NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

**THE DDD-III: A RESEARCH PARADIGM
FOR ABSTRACTING JOINT C3 SCENARIOS
FOR TIER-1 EXPERIMENTS.**

by

Gregory Scott Higgins

June, 1996

Thesis Advisor: David L. Kleinman

Approved for public release; distribution is unlimited.

DTIC QUALITY INSPECTED 3

19960910 147

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE June 1996	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE THE DDD-III: A RESEARCH PARADIGM FOR ABSRACTING JOINT C3 SCENARIOS FOR TIER-1 EXPERIMENTS		5. FUNDING NUMBERS	
6. AUTHOR(S) Higgins, Gregory S.			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey CA 93943-5000		8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSORING/MONITORING AGENCY REPORT NUMBER	

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

Advances in communications technology and computers have made possible tremendous leaps forward in real-time Command and Control (C2). This revolution in C2 capability will provide decisionmakers (DMs) in the Joint military organization with an unparalleled tactical and strategic picture of the battlefield ("Global Awareness"). The ways in which DMs having Global Awareness coordinate their information, resources and activities to fulfill the organization's mission is the focus of the Adaptive Architectures for Command and Control (A2C2) project. In order to examine these command and control issues empirically, the A2C2 project required a multi-player real-time simulation environment. A new computer model was needed to abstract "real world" problems into a controlled laboratory environment. The result was the Distributed Dynamic Decisionmaking (DDD-III) paradigm. The phase one experiment of the A2C2 project was designed to validate the DDD-III paradigm, with emphasis on the manipulation of organizational variables. This document reviews the project objectives, DDD-III capabilities, experiment one scenarios and scenario development issues. The scenario generator users guide and players tutorial, developed during phase one, are provided. The intent of this document is to link the phase one experiment to the next, more advanced phase of the A2C2 project.

14. SUBJECT TERMS Distributive Dynamic Decisionmaking Adaptive Architechtures	15. NUMBER OF PAGES 185
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

DTIC QUALITY INSPECTED 3

ii

THE DDD-III: A RESEARCH PARADIGM FOR ABSTRACTING JOINT C3 SCENARIOS FOR TIER-1 EXPERIMENTS.

Gregory S. Higgins
Lieutenant Commander, United States Navy
B.E. E. Villanova University, 1985

Submitted in partial fulfillment
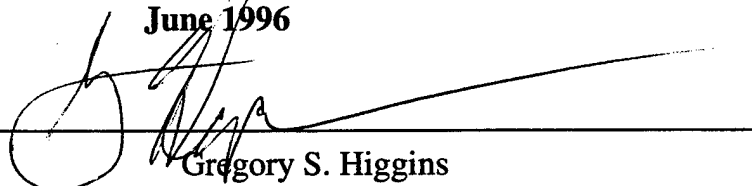of the requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS TECHNOLOGIES
(SPACE OPERATIONS)

from the

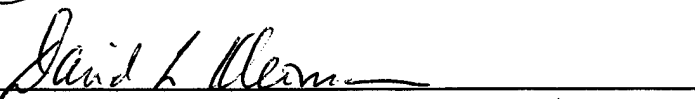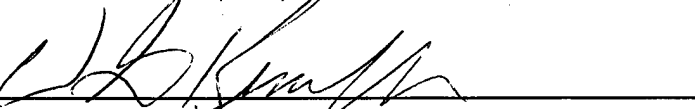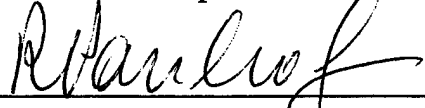## NAVAL POSTGRADUATE SCHOOL
June 1996

Author: _____
Gregory S. Higgins

Approved by: _____
David L. Kleinman, Thesis Advisor

_____
William G. Kemple, Second Reader

_____
R. Panholzer, Chairman
Department of Space Systems Academic Group

iii

ABSTRACT

Advances in communications technology and computers have made possible tremendous leaps forward in real-time Command and Control (C2). This revolution in C2 capability will provide decisionmakers (DMs) in the Joint military organization with an unparalleled tactical and strategic picture of the battlefield ("Global Awareness"). The ways in which DMs having Global Awareness coordinate their information, resources and activities to fulfill the organization's mission is the focus of the Adaptive Architectures for Command and Control (A2C2) project. In order to examine these command and control issues empirically, the A2C2 project required a multi-player real-time simulation environment. A new computer model was needed to abstract "real world" problems into a controlled laboratory environment. The result was the Distributed Dynamic Decisionmaking (DDD-III) paradigm. The phase one experiment of the A2C2 project was designed to validate the DDD-III paradigm, with emphasis on the manipulation of organizational variables. This document reviews the project objectives, DDD-III capabilities, experiment one scenarios and scenario development issues. The scenario generator users guide and players tutorial, developed during phase one, are provided. The intent of this document is to link the phase one experiment to the next, more advanced phase of the A2C2 project.

TABLE OF CONTENTS

I. INTRODUCTION

A. DISCUSSION

Rapid advances in communications technology and computers have made possible tremendous leaps forward in real-time Command and Control (C2). This revolution in C2 capability will soon provide decisionmakers (DMs) in a Joint military organization with an unparalleled tactical and strategic picture of the battlefield. Additionally, a decisionmaker will have the ability to rapidly access all information available within the organization, monitor actions made by other DMs, and call upon a world-wide data base to aid in the decision process. This powerful, all-knowing capability has been dubbed the name "Global Awareness".

The processes by which DMs having Global Awareness coordinate their information, resources and activities to fulfill the organization's mission, (especially in a dynamic and uncertain environment,) are adapting in response to the changing informational infrastructure. Current research involving adaptive architectures for Joint Command and Control (C2) seeks to examine the interactions between mission requirements and organizational structure. The Office of Naval Research has commissioned a research effort into this far reaching topic - the Adaptive Architectures for Command and Control (A2C2) project. This project will explore the changes in task structure that drive changes to organizational structure in the Joint warfare environment, and how the Joint organization should adapt to such changes.

In order to examine these command and control issues empirically, the A2C2 project required a multi-player real-time simulation environment capable of :

- Focusing on the dynamic/execution phases of the mission
- Allowing easy manipulation of key structural variables in task and organizational dimensions.

The approach taken combined empirical and analytical efforts to develop a model–based program of experimentation with human teams. A first step in this process was to abstract "real world" problems in order to bring them into a controlled laboratory environment where we could manipulate a large variety of experimental conditions.

A new paradigm was needed to support such model-based empirical research, resulting in the development of the Distributed Dynamic Decisionmaking (DDD-III) paradigm. This third generation paradigm is an evolution of an earlier paradigm, the DDD-II, which was the backbone of extensive empirical research from 1989 - 1995 involving "open-ocean" naval team (distributed) decisionmaking and coordination.

This thesis will overview the objectives of the A2C2 project and the general requirements of the tier-1 experiments conducted 4 - 16 March 1996 at the Naval Postgraduate School. Emphasis will be on the operationalizing of various dimensions of task and organizational structure and the new DDD–III paradigm.

B. APPROACH

1. Project requirements

We will begin by reviewing the A2C2 project goals. Chapter II will cover the general requirements of the project, with emphasis on interactive computer simulations - what is currently available and what options were considered for phase one. The basic framework of the DDD-III will be discussed, as well as the environment contained in this new simulation.

2. Core elements

Once the basic DDD model is defined, chapter III will move further into the paradigm and a description of the core elements. What elements can the scenario designer control to investigate an overall hypothesis? Broken down into three general areas; task, platform and organization, the chapter will discuss the specific concepts behind each variable changed and the measures monitored to view the results. A description of the various dimensions of organizational structure contained in the DDD-III will be reviewed.

3. Phase I experiment scenario

Beginning in chapter IV the focus of the thesis will shift from the generic DDD applications to the specifics of the phase one experiment conducted in March 1996. The purpose of this experiment - to test interaction between organizational structure and task

structure, will be reviewed. The general hypothesis, implementation and abstraction will be explained, emphasizing the details of the scenario used.

4.   Simulation Development

Chapter V will cover development issues that arose during the phase one experiment. The DDD-III continued to evolve during the phase one scenario design, allowing for abstraction of specific missions and actions in the Joint world. The challenges faced in the design of the display interface, the movement of the various assets and the representation of threats are outlined here - along with the solutions found.

5.   Future Issues

Finally, chapter VI will address several scenario and software elements that arose during phase one but were determined to be future issues.

6.   Development aides

Appendix A contains the scenario generator users guide, developed to assist future experimenters in the design of scenarios and their abstraction from the real-world to the DDD world. This users guide, when coupled with the players tutorial contained in Appendix B, represents the link to phase two experiments.

II. A2C2 PROJECT REQUIREMENTS FOR THE DDD-III PARADIGM

A. A2C2 PROJECT DESCRIPTION

The Adaptive Architectures for Command and Control (A2C2) project is a four year research effort commissioned by the Office of Naval Research (ONR) to study emerging issues in command and control (C2). The project's overall goal is to increase the level of knowledge regarding decisionmaking in organizations that must operate in an environment, or mission context, that is subject to change. A secondary goal is to extend 12 years of composite naval warfare decisionmaking research into the Joint C2 arena. The focus of the A2C2 project is centered on adaptive architecture structures, such as authority, communication, resources ownership, etc., and specifically how these architectures should be driven by changes in the Joint environment.

The A2C2 project involves an "industry-university-government" initiative, with researchers from the Naval Postgraduate School (NPS), the University of Connecticut, Alphatech, Inc., George Mason University, and the MITRE Corp. The research program is a three-pronged, coordinated effort that involves field, experimental, and theoretical components:

- Interviews with experienced joint officers

- Participation in exercises and demonstrations (pooling of theoretical and analytical techniques to provide models of decisionmaking and adaptation)

- A three-tiered series of experiments with officers in joint settings for measurement of individual and team performance in dynamic, evolving missions and scenarios. The intent is to subject the results from tier-I experiments to further scrutiny in tier-II experimentation, and ultimately to apply them to improve the command and control abilities of actual C2 operational organizations.

As part of the charter of the A2C2 project, a tier-I experiment to test hierarchical structure was conducted by the research group at the Naval Postgraduate School (NPS) in March 1996. This experiment represents phase one of the A2C2 project.

B. SIMULATIONS

In order to determine empirically how structural changes effect an organization's command and control process it is necessary to be able to control the many variables that describe that organization. There are basically two paths that can be taken to conduct controlled experiments. The first path, which involves the use of real-world assets and command organizations to investigate an experimental hypothesis, is unrealistic. It is fair to say that it would be "difficult" to convince the Department of Defense to "loan" a fully operational Joint Task Force to the A2C2 project for the purposes of experimentation. The second, more practical, method is to abstract the important elements of the real-world command and control problem into a computer model, or simulation. This simulation can then be administered to test subjects, representative of a cross-section of the general

military officer populace. The performance of the decisionmaking teams in this simulation can then be analyzed statistically, and the results extended to the real-world.

1. Other simulations

Computer simulations are not a new concept. Many attempts have been made to model the warfare environment, some with more success than others. Several excellent war gamming simulators exist today, such as RESA, JTLS and MTWS. These are used by individual services and Joint commands to train for future conflicts. These models allow for high-level, theater emulation of tactics, but in our experiment the tactical performance of the team being tested was in itself not important. The unique A2C2 project requirement - to change task and organizational structure, thereby directly influencing team performance, - did not appear to be effectively supported by any of these simulations. Each was reviewed by the A2C2 project, not just for use during phase one but for future use as well.

a. Research Evaluation System Analysis (RESA)

RESA is a high-level Naval war game which is supported by an extensive secret data base and tactical outcomes matrix. The ability to move above the individual unit and component level to Fleet and theater decisionmaking is supported, although RESA is not designed specifically for that task. The data collection abilities of RESA are actually quite good, with some user definition possible and most measures of

effectiveness (MOE) automatically recorded. The scenarios contained in the software can be modified fairly easily, which would allow some degree of control over the tasking and asset competition of the team members. However, operationalizing the changes in organization (and task) structure would be more by accident than by design - one would have to "trick" the computer into chain of command changes. The ease of use and the relatively short training period required for test subjects would be an advantage in cycling a large number through the simulation in a short period of time. RESA, however, has no capability to model amphibious or ground warfare components, making this simulation poor for A2C2 needs in modeling Joint operations. Most significantly, any changes made to the simulation could not be supported without major revisions in software.

b.   *Joint Theater Level Simulation (JTLS)*

JTLS is a high level Joint war game, supported by a large computer data base of every conceivable tactical combination of weapons, threats and maneuvers. This simulation models the theater level decisionmaking organization while simultaneously requiring the user to specify such items as individual platform weapons loads. This combination of high level decisions and low, low level planning makes JTLS completely unwieldy and cumbersome for our purposes. The simulator is excellent in the representation of ground forces, weak in the amphibious world and very, very weak in the Naval world. As a result of these combined deficiencies is would be extremely difficult to accurately model the Joint world. (The name of the simulation not withstanding.)

8

Overall, the simulator is not suitable for abstracting the changes in organization structure required. The scenarios and command structures cannot be easily changed by the user and would require enormous software modifications to provide the required flexibility. Additionally, these software changes would have to be made through an administrative and maintenance contractor who holds the rights to the software.

c. *MAGTAF Tactical Warfare System (MTWS)*

MTWS is an excellent, high-level, <u>Joint</u> war game supported by an extensive data base, tactical matrix, and outstanding theater level command structure. The simulation is mainly concerned with the Theater level decisionmaking process. Data collection abilities of MTWS are outstanding, with user definition possible and multiple MOE's covered. The scenarios contained in the software can be easily modified, with all degrees of tasking and competition supported. The ease of use and short training period required for test subjects, coupled with the small number of support personnel would make it possible to have a high rate of experiment throughput. If MTWS had been available for <u>small scale use</u> at NPS, tier-I experiments for the A2C2 project might have been run using that simulator. Tier-II experiments using MTWS are a very real possibility. Providing the ability to incorporate players not collocated, MTWS may soon be contained in the Global Command and Control System (GCCS) currently coming on-line in the field. The GCCS, in addition to possibly supporting the MTWS package, provides the decision teams with Distributed Collaborative Planning Aids, a tool that will

be used by Joint planners in the real-world. It is envisioned that a candidate for A2C2 tier-III empirical studies will be GCCS with MTWS as a Joint hybrid game.

2.   Distributive Dynamic Decisionmaking (DDD) -III

After reviewing the simulators currently in use and available it became apparent that a user friendly, easily modified, "low level" command and control simulator was required.   To support this need the Distributed Dynamic Decisionmaking (DDD-III) paradigm was developed, based upon the earlier DDD-II paradigm [Ref 1], which was the backbone of extensive empirical research from 1989 - 1995 involving "open- ocean" naval team (distributed) decisionmaking. The DDD-II paradigm, adaquate for its designed purpose, allowed for very little manipulation of command structure and other organizational variables. The DDD-III represents a huge leap forward in one's ability to model the Joint real-world environment, with a wide range of commands and user defined parameters available to abstract the complex components and variables associated with a Joint battle space.

DDD-III provides a multi-player, real-time, model-based simulation environment to conduct empirical research within a controlled laboratory environment. The design of DDD-III focuses on the dynamic/execution phase of the mission and allows for easy manipulation of key structural variables both in task and organizational dimensions.

10

The software has the ability to handle many abstractions of the Joint world, including:

- Geography/distance

- A wide variety of task types and classes

- Moveable platforms (with sensors)

- Weapons (resources)

- Subplatforms

- Ability to impose communication constraints

- Ability to manipulate the information/resource structure

The DDD-III was developed to abstract the decisionmaking environment faced by a Joint military organization. Limited resources must be allocated by human decisionmakers (DMs) to prosecute tasks (i.e., activities or "things to do") in a dynamic and uncertain environment. Command and control issues are introduced to the simulation exercise though changes in organizational, task and information structures. Objective (verses subjective) data collection exists at all levels of the simulation and can be tailored by the scenario designer to record nearly every possible variation of the team's performance. While not a true tactical model, the DDD-III can be conceptualized as a bridge between the real-world Joint (operational) environment and the laboratory.

3. Objectives of the DDD-III

The overall objective of the DDD-III development was to provide a multi-player, real-time environment to support tier-I (controllable, laboratory-based) empirical research for the A2C2 project. The DDD-III was designed to specifically fulfill the following needs:

- Create a "realistic" Joint environment within a computer simulation, abstracting the various elements of Army, Navy, Air Force and Marine warfare, found in the Joint Task Force environment, to an understandable computer representation.

- Provide for the easy manipulation of key structural variables to allow testing of basic hypotheses dealing with structural change.

- Avoid the need to build/require a technical domain expertise in test subjects, thereby allowing the experiment to be conducted on-site at NPS using available officers.

- Create a simulator that would allow for ease of scenario design/change, data collection and data retrieval.

C. THE DDD-III ENVIRONMENT

The overriding hypothesis of the A2C2 research is that changes in task structure will drive changes to organizational structure. Based on this, the DDD-III scenario is crafted by the experiment designer to uncover key issues in the organization's

decisionmaking and/or coordination processes. The DDD-III must therefore have the ability to explicitly operationalize many of the relevant organizational dimensions within a Joint task force (JTF). There are three essential factors that make the DDD-III ideal for the tier-I experiments to be conducted by the A2C2 project. They are: the concept of military "Jointness", the ability to abstract mission level requirements and finally, to impose competition for assets on the teams being tested in order to force intra-organization coordination

1.   Joint warfare framework

Warfare of the future will be conducted using mobile, flexible military units requiring streamlined command structures. It is almost certain that the military of the future will be smaller and that all major operations will be Joint in nature. Based on these assumptions, the Joint warfare concept was used to frame the DDD-III paradigm. A Joint Task Force (JTF) commander and subordinate component commanders are used as the base level model. Each team member, or decisionmaker (DM), is responsible for different but overlapping elements in a common battle space. The forces available to the DMs are defined not by the individual service, such as Army, Navy, etc., but by the mission to perform; i.e. land, sea, air.

2. Mission structure

The DDD-III scenario requires test subjects to follow a predefined operational order (OPORDER), while simultaneously defending one or more "penetration zones" and their own assets, against potential land, sea, and air threats. The OPORDER specifies the mission, which is represented in the DDD-III as a series of "tasks". Tasks are defined as abstracted activities (things to do) and can range from taking a beach, clearing a minefield, or identifying and (if necessary) prosecuting a hostile contact. The scenario designer can create a template of tasks, with defined attributes, threat capabilities and movements. Each task template can then be linked together by time, space and/or precedence to provide a theater level <u>mission</u> for the organization to accomplish.

3. Force competition

Competition over assets is a major issue in the real-world Joint environment. This competition is used to "force" coordination among the DMs - they must set priorities, adjusting them dynamically to allocate limited organic and non-organic resources in such a manner as to achieve the *overall* mission objective. How teams coordinate their efforts, under various organizational structures is the focus of this experiment. The competition for assets is effected through manipulation of the task structure, task assignment and responsibility, the command structure and communications structure of the decisionmaking team, etc. The DDD-III allows the scenario designer to give each DM responsibility for a specific mission area (set of tasks), as defined by

OPORDER. The DMs must take action to maneuver their assets/platforms so that tasks are brought within sensor or weapons range. As the scenario unfolds, a DM who is prosecuting tasks with his own platform(s) might require additional assets from other units within the JTF. Thus, the team must coordinate the allocation of assets and activities, both for information gathering and task prosecution. Limiting or constraining the assets available to the organization will cause DMs to compete for assets in order to accomplish their *local* objectives. This is especially pertinent if the scenario has been designed wherein several DMs require a resource simultaneously. These scenario design abilities make the DDD-III a powerful empirical instrument.

III. DDD-III CORE ELEMENTS

The DDD-III software gives the user the ability to modify dimensions of task and organizational structure in order to test a specific hypothesis. These dimensions are considered to be the core elements of the paradigm. We manipulate these dimensions to study their interactions, and to present to an organization those task conditions that we believe will elicit the desired changes in organizational structure and response. The dimensions that can be manipulated are:

- Task (threat/mission/enemy forces)

- Platforms (own forces)

- Organization (command and control)

A. TASK AND TASK STRUCTURE

DDD-III represents all missions, opposing forces and threats as "tasks". During the play out of an experimental run, tasks appear, move/maneuver, and disappear according to the designer's scripted scenario. The designer has the ability to define various dimensions of task structure in order to closely align the DDD-III scenario with the overall mission (OPORDER) of the JTF. The characteristics of each task; its class, attributes, and resources required, are tailored to specify the threat (such as a fighter, minefield, etc.), and to impose situations that will create intra-team coordination (for the assets necessary) for task prosecution.

17

1. Task type and class

Tasks are objects, categorized as one of three types: air, surface, or ground. Within each type, tasks are further divided into unique classes according to scenario requirements. This allows the experiment designer to create, within a given task type a variety of classes. Examples of ground type task classes are: minefield clearance, armored columns, taking a ground area, surface-to-surface/SCUD missile sites, and even "false" tasks. The only real limit to the representative classes is the designer's imagination - the DDD provides a great amount of flexibility. Each class of task/activity can be assigned an initial priority, which DM has responsibility, and who has the authority (or ability) to prosecute the task(s).

2. Task attributes

Associated with each task is a set of (numerical) attributes, written as a vector $(A = [a_1, a_2, ..., a_n])$ that defines the various characteristics of the task quantitatively. These attributes are used to specify such elements as speed, weapons potential, evasive ability, IFF status, etc. The number of attributes and their definition depend only on the specific needs of the problem being studied. (The first two attributes of any task are in fact fixed in definition: Value and Attack time) The true attribute vector for each task within a given class is drawn from a normal distribution with a specified mean and standard deviation. The DDD-III software provides the designer the option to tailor/override the individual values of attributes on a task-by-task basis.

Sensors on board the various platforms obtain measurements of task attributes, provided the tasks are within their sensor range. The DDD-III has the capability to apply a level of "noise" and "bias" to these attribute measurements, providing the ability to obscure the true values from the DMs. If the attributes are needed to identify task class (e.g., to discriminate a hostile from a neutral, or to identify the class of an incoming patrol boat), this obscuring of the measurements may result in mis-identifications, a situation that is very possible in the real-world. The means and standard deviations of the sensor noises and values of the (circular) sensor ranges can be functions of both task and platform class.

3. Task attributes-to-resource mapping

The attribute vector gives the DMs requisite information regarding a task's characteristics. Once a specific task has been identified or classified, the DM must determine the assets required to prosecute it. Associated with each task is a resource vector ($R = [r_1, r_2, ... , r_m]$) that defines the resources required for a successful prosecution. The designer-defined elements of R can be viewed as generic weapons requirements, such as ground suppression, mine clearing, anti-air capability, etc., depending on the abstraction level of the simulation. Default values for the r_i are generated within the DDD-III via a designer-defined mapping function $f(A, ID)$. This function generates resource requirements from the attributes of the task and its class. The DDD-III software also allows the scenario designer to override these values and give a

19

specific resource requirement to any individual task. In the play of the game, subjects are automatically given the results of the attribute-to-resource mapping based on their current estimates of A and class ID. This process mimics a staff function recommendation or a decision support system. Imprecise knowledge of a task's attribute values and/or a mis-identification of class ID will therefore result in incorrect estimates of required resources by the DM.

Attributes and resources are the DDD's lowest-tier constructs for representing information/data and weapons systems. We believe that these are valid constructs for hierarchical aggregation of activities and assets. This set of constructs precludes the experimenter from having to build scenarios from the "bottom-up" to study problems at a large force/component level. By treating task attributes and task resources as two separate vectors, the DDD-III can be used to study pure (distributed) information processing problems, pure (distributed) resource allocation problems, or hybrid cases. For example: in a pure resource allocation context, the attributes can be associated one-to-one with the resources ($r_i = a_i$), with the "measurements" of the a_i being made noise-free. Our first experiment with the DDD-III used just such a construct.

4.   Task precedence/prerequisites

A crucial element in scenario design is the ability to assign prerequisites (or corequisites) to task accomplishment. This is an important dimension of task structure, as it defines correlations and coordination requirements among the individual activities

20

that combine to comprise the mission as a whole. The earlier DDD-II considered each task as an independent threat (a "mosquito"), with no consideration given to tasks not considered threats or potential threats . The precedence structure implemented in the DDD-III allows for task fan-outs and fan-ins. With respect to task fan-outs, the lack of, or delay in, accomplishment of a single task can inhibit action on many tasks. For example: taking a high ground can be a prerequisite for follow-on introduction of forces. With respect to fan-ins, a number of diverse activities may require completion prior to mounting a final attack, such as on an airfield (which may be one of the mission objectives).

A "real" task environment does not usually inhibit a DM from making a decision or taking an action. In such cases, if actions are completed out of sequence and deviate from the initial plan, the consequences may be severe. In our empirical research we need the ability to control task structure to a sufficient degree as to prevent teams from creating totally arbitrary paths to achieve a final objective. The specification of a prerequisite mission/task structure (e.g., clearing minefields before landing on a beach), along with the DDD-III's forcing adherence to this structure, provides the appropriate degree of task control.

5.   Task parallel processing

Parallel processing of a task by several DMs is now supported in DDD-III. This is a key element in scenario design when assets owned by different organizational units

are to be coordinated in time/space for a simultaneous attack. An assault on a hill by a Marine unit that is simultaneously supported by CAS, artillery, and Naval gunfire is an example of such a multi-resource activity. The DDD-III "scores" the effectiveness of the attack as a function of the sychronicity (and correctness) of the allocated assets.

6.   Task spawning

Missions in the real-world often develop secondary mission objectives during prosecution. In DDD-II, task arrivals and disappearances were solely time-driven. No consideration was given to event-driven task arrivals. In this environment it is difficult to script enemy actions in response to the actions of the subject team. To compensate for this short coming, DDD-III was heavily modified to include enemy counterattacks, medical evacuations, and other tasks that would be "spawned" by a specific action. (Such as a beach landing). The DDD-III implements an extended task structure by allowing a primary task to spawn secondary tasks either upon attack of the primary task, or upon disappearance of the primary task. Spawning allows the scenario designer to provide dilemmas and conflicts for the organization's decisionmakers. This feature also prevents a rote scenario, with the team being led down a single path, solely for the sake of the hypothesis under test. For example, the failure to identify and prosecute an enemy submarine by a specified time would result in a cruise missile attack upon the CVBG, thereby creating a new defensive requirement for the team.

7.   Task removal

Often the existence of a specific mission is linked to another mission being pursued, whether by own forces or another DM. For example, the complete destruction of an artillery company by tactical aircraft would negate the requirement for Naval Surface Fire Support (NSFS) to suppress this position in the future. In DDD-III, tasks can be linked so as to allow for this possibility. The DDD-III implements a task removal structure by allowing a primary task to "cancel" a number of secondary tasks upon the destruction of the primary task. Task removal allows the scenario designer to further increase the richness of the simulation, providing the DMs with a real-world, responsive mission structure as the team progresses through the scenario.

B. PLATFORMS AND PLATFORM STRUCTURE

A "platform", or asset, is a basic element of the DDD paradigm that carries sensors and resources, modeling the teams friendly order of battle elements. Examples of platforms are ships, helicopters, ground units, bases, etc. A platform may also carry subplatforms. For example, a carrier can contain helicopters and various fixed-wing aircraft, and the helicopters can carry sonobuoys, etc. With this nesting ability a platform/asset structure can be tailored to any level of detail. An amphibious landing, for instance, can be modeled by a platform-subplatform structure using sea-going platforms that carry ground units for launch at a shoreline. These ground units may then have sub-units of their own.

The team identifies and prosecutes tasks by allocating/scheduling its platforms, ideally to make best use of their generic sensors and resources. A platform can be controlled (i.e., geographically repositioned, commanded to attack, etc.), only by the current platform owner. Assets have sensor and weapons' ranges, which generally depend on task type and platform class. A new feature in the DDD-III is that each platform now has a "be-attacked" range, within which specified task classes can inflict damage to it. The values for all ranges can be manipulated via the paradigm depending on the experimental requirements.

1. Platform type/class definition

In a manner similar to tasks, platforms are categorized first by type: air, sea or ground, and then by class. The number of platform classes and their individual characteristics depend on the requirements of the experiment. Nearly any military platform can be represented in an abstracted manner, limited again only by the designers imagination. All platforms of a given class will have the same parametric features with respect to sensors, weapons, maximal velocity and available subplatforms. The only difference among platforms within a given class is their initial owner and location.

2. Subplatforms

Subplatforms are actually platforms located on board another, parent, platform. These subplatforms provide the DDD-III with a more accurate way to model real-world

assets. The "nesting" of subplatforms allows for an additional element in the hierarchical asset structure. A subplatform does not become an independent platform until it is "launched" from the parent platform. The DM who owns the parent may launch subplatforms that will become available after a specified launch time delay. The DM is limited to those subplatforms available on the parent platform, and may only launch one subplatform class at a time from each platform. Ownership of the subplatform "child" can be specified independently. For example, VF assets onboard a Carrier (which is owned by the CVBG commander), could be owned by the JFACC when they are launched.

A subplatform is only effective (available for use) for a limited time period. Subplatforms are either returnable to their parent (such as helicopters), or non-returnable (such as sonobuoys, missiles, etc.). Once returned, a subplatform is not available again until a recycle time has elapsed. Subplatforms can also be designated as reusable or non-reusable. An asset that is non-reusable can only be used to attack once, (such as an artillery round) whereas a reusable subplatform can attack multiple times, within its availability window. All parameters relative to subplatform structure and availability are designer specified in the DDD-III.

3. Platform sensors

Each platform has three types of generic sensors to obtain information on air, surface, and ground type tasks. These sensors have specified effectiveness ranges,

modeled as circular regions, for task detection, measurement, and identification/classification which depend on platform class. However, the DDD-III allows the designer to fine-tune these ranges according to individual *task* classes. Thus, it is possible to implement an engineer platoon that detects mines at a further range than a tank column would. For a platform to "see" a task, it must be within the platform's detection zone; to obtain a measurement of task attributes, the task must be within the measurement zone; to obtain task class identification, the task must be within the classification zone. As noted earlier, measurements of task attributes may be "contaminated" by noise.

4. Platform resources

Platforms also contain weapons (or general resource capabilities) for processing tasks. The resources on an individual platform (or subplatform) are defined by a generalized resource vector, or vector of generalized combat capabilities, $R = [r_1, r_2, \ldots, r_m]$, where the elements r_i are the same as those associated with task processing requirements. A platform of a given class can be used to attack any task, but the range of the platform's weapons is a function of task type (or class). In order to engage a task, a DM must move one or more selected (sub)platforms within range of the task for (identification and) attack. Once an attack starts the platform(s) are tied-up for a length of time equal to the task's processing time required, a_2.

The resources on the platforms assigned to attack a task must meet or exceed those required by the task for the attack to be rated as successful. If the summation of allocated resources, on an element-by-element basis, is less than the required amount, the attack will achieve partial success at best. By giving a specific task a value of R the DDD-III establishes what (mix of) assets with their corresponding Rs suffice to correctly process that task. In this vein, platforms can also be made job-specific. For example, mine-clearing helos could have values for, say, r_5 corresponding to those required for mine clearing tasks, but other assets would have lower values of r_5, or zero.

The expression for determining the accuracy of the attack on task j, P(j), can be user-modified. After each attack the DDD-III gives a "gain" to the team of V(j)*P(j) and a "loss" to the team of V(j)*[1 - P(j)], where V(j) = a_1 = task value. As noted earlier, the resources required for task processing are a function of task attributes. The DDD provides each DM with an estimate of resources required, using his/her current estimate of task attributes and class ID. Thus, accurate task information processing is a precursor to correct resource allocation. For example, attacking a task defined as a non-threat will always give P(j) = 0.0.

There is no attrition in the current implementation of the DDD. A poorly done attack, an enemy penetration of a defense zone, etc., result only in point loss, and not in a loss of assets or asset capability. This was done so that the resources available are not a (uncontrollable) function of the team's sample path through the scenario. This is critical to controlled experimentation since the task/mission structure is generally predicated on

27

asset availability. If task structure is to be an independent variable, the total assets available should remain constant, or else the team should be allowed to restructure the mission.

C. ORGANIZATIONAL AND ORGANIZATIONAL STRUCTURES

Organizational dimensions refer to the ways in which a multi-person organization is structured. These dimensions include authority, resource, information and communication structure. The structuring of a large organization must maintain a general congruence with the task structure to assure that DMs assigned to subtasks have the resources and information required (or can obtain them via a request chain), can communicate with other DMs with whom they need to coordinate, and that the "chain of command" that is established facilitates successful mission completion. Organizational design is a multi-dimensional problem, where one dimension cannot be changed without considering concomitant changes in other (supporting) dimensions. The dimensions supported in the DDD-III are described below.

1. Command/authority structure

Military organizations adopt a command hierarchy to structure the overall decisionmaking process. The DDD-III allows for tailoring a "chain of command", provided that each DM in the organization has no more than one "boss". We use a general acyclic tree structure to code the command hierarchy by giving to each DMj a

single integer $c(j) \leq j$ that defines to whom DMj reports. Command structures ranging from totally disconnected [$c(j) = j$], to totally flat [$c(j) = 0$], to totally serial [$c(j) = j-1$] can be treated in this construct. For example, in our experiment wherein two units reported directly to the CJTF (DM0), and two other units reported to a common mid-level functional commander (DM1), we had C = [0,0,0,0,1,1].

This generalized authority or command structure is operationalized via task assignment and platform/asset assignment capability. As only those DMs assigned to a task are allowed to prosecute it, a leader can control dynamically the coordinated actions within his (sub)team, and reassign platforms according to mission need.

a.   Task assignment

In the DDD-III a DM can (re)assign or co-assign a specific task to any subordinates DM in his/her sub-organization (subtree). Assignments cannot be made upward nor laterally. Assignments are constrained in that: i) a task cannot be taken on unilaterally, i.e., the task must have already been assigned to someone in the subtree, and ii) the task cannot be given away unilaterally, i.e., assigned away from all DMs.

b.   Platform assignment

The DDD-III operationalizes a DM's authority over the assets in his/her sub-organization in two ways. He can *advise* a subordinate DM to transfer his asset(s) to another DM, or he can independently and unilaterally *force* a transfer action. If an asset is

owned by someone outside of his sub-organization, then all that a DM can do is to request use of that asset (from the owner or via the chain of command).

2.   Resource access structure

This defines both platform ownership and the "rules" by which platforms can be requested, transferred, and accessed. At any given time each platform is controlled by the DM who owns it. If a platform is defined as transferrable, it can be transferred during the simulation from one DM to another, either by the owner or by any of the owner's superiors acting through the chain of command. Other platforms can be defined as non-transferable, e.g., the amphibious ships (which belonged to the ARG), and the aircraft carrier (which belonged to the CVBG) in experiment 1. The transfer of an asset from one DM to another requires a finite (user-specified) time delay, during which period the asset is "locked up," or inhibited from accepting other commands.

A subplatform nesting structure, wherein the "owner" of a platform is not necessarily the owner of its subplatforms, is supported within the DDD-III. Some subplatforms can also be assigned permanently to the parent asset irrespective of ownership, such as a ship's self-defense assets.

3.   Information structure

The information (access) structure describes how data collected by platform sensors is distributed to the various DMs in the organization. The information structure is

operationalized in the DDD-III by defining an information network. Each DM can be assigned a level of "tie in" to the information network depending on task type. Each task class can be tailored to be viewed differently by each platform class, allowing for a truly robust and diverse information architecture.

For example, a leader (who is responsible for global coordination) can be provided with global information, while other DMs can be given more detailed local information depending on their local decisionmaking responsibilities. A centralized, partially centralized or decentralized information structure can be created by setting the different network "tie in" levels by task/platform class for different DMs. In the first experiment a common operation picture, or true Global Battle Space Awareness, was used. Each DM was able to see in real-time the location of all contacts detected in the task force operations area. However, detailed attribute data needed for resource allocation and/or classification was sometimes only provided to the local DM assigned to that task.

4.   Communications structure

The communication structure specifies who can send messages to whom, and also includes parametric data such as receipt delays, equipment delay and "bandwidth". By preventing communications between certain units and forcing communications up and down a specific chain of command we are able to observe the impact of different

communications structures not just in isolation, but more interestingly in its relationship (interaction) with other structural dimensions.

In running the DDD software we generally inhibit verbal exchanges among DMs in order to ease subsequent analysis of the communications data. Thus, computer-mediated communications is the only way by which DMs can share local information about task attributes and ID, request assets, and coordinate actions. The communication among DMs is effected through the graphics user interface via a set of preformatted commands:

- Request information: ask a DM to send his local information about a task.

- Transfer information: transfer one's local information about a task to another DM.

- Request platform: ask a DM to transfer the ownership of a platform to another DM.

- Transfer platform: inform another DM that a platform is being transferred to him (or that an earlier request is being denied).

- Coordinate action: several choices are available in this command and can be easily changed to accommodate specifics of the simulation:

  - Ask another DM to either handle, support, or ignore a task.

  - Tell other DMs of one's intent to handle, support, or ignore a certain task.

Copies of all messages sent by a DM (except for information requests/transfers)

are automatically sent to that DM's superior in the chain of command. This keeps the higher-level DM apprised of the needs/activities of his subordinates, and is an attempt to capture overheard (open) message flow on a DM command net.

To represent communication and data processing delays, the DDD places a time delay on a message transfer. To model a limitation on channel capacity (or channel access), the number of communications (N) in a fixed time window (T) can be specified.

D. DATA COLLECTION

The first three core elements of the DDD-III were dimensions to be manipulated to pursue the experimental hypothesis. The last element and possibly the most important is that of data collection. As stated earlier, one of the critical requirements for our simulation paradigm was useful data collection and retrieval. Data collection can be broken into two subject areas - the data to be recorded, or measures; and the method by which the computer system records this data.

1. Measures

The A2C2 experimental scenario designer needs the ability to specify exactly what measures would be recorded and in what format these measures would be presented. The measures for the tier-I experiments can be broken down into two major groups; performance and process.

a.   Performance

Measures within the performance category represented the overall scores of the decisionmaking team in regards to mission accomplishment and team strength. The emphasis here was in accomplishing the task, not in <u>how</u> it was done. These elements included team strength and the time it took for the accomplishment of tasks or series of tasks.

b.   Process

Measures within the process category recorded the mechanisms by which the team accomplished the missions. This set of measures was critical in analyzing the data in regard to the overall experimental hypothesis. These measures can be further broken down into two broad areas:

- Communications - A critical measure of the tier-I experiment was the interaction between the lower level commanders and the command hierarchy. This set of measures indicted communications pathways, time of replies and method of requesting assistance.

- Decisionmaking - There were several measures that indicated the teams ability to solve problems, such as competition of assets, and how well the team was working together. One example of this type of measure was the competition score. It was expected that this

measure would be an indicator of the degree to which the intermediate level of hierarchy contributed to resolving, or reducing competition [Ref: 3].

2. Output files

The ability of the DDD-III to provide rapid data retrieval capability was essential to our experiment. The software was designed to present data in two separate files; the Log file and the Dependent variable file (dep file).

a. *Log file*

The log file is a record of all actions that occurred within the scenario. The file is time-tagged and can be used to replay the entire scenario. This feature is of tremendous benefit to the experimental designer, who, at a later time, can designate additional measures and re-run the scenario to recollect the new measures.

b. *Dep file*

The Dep file is defined by the experimental designer prior to the scenario run. The content of the Dep file can be a direct report of measures or an aggregation. Available immediately following a scenario run, the Dep file provides the ability to perform analysis on-site.

IV. DDD-III EXPERIMENT PHASE I SCENARIO DESCRIPTION AND REQUIREMENTS

The DDD-III software provides a mechanism to empirically investigate proposed Command and Control theories in a laboratory environment. For the application to the A2C2 project, the general elements of organization, assets and threats discussed in the first three chapters were used to abstract a specific Joint warfare simulation. In this chapter the specifics of the phase one A2C2 experiment will be covered.

A. EXPERIMENT ONE HYPOTHESIS

The initial A2C2 experiment was designed to investigate the general hypothesis that there is an interaction between task structure and organization structure. As stated earlier, the context for this experiment was that of a Joint task force, involving or not involving the use of common functional commanders. For our purposes then the hypothesis can be summarized more effectively as follows: "An organization with a common functional commander is better for certain tasks than an organization without one." The goal of this phase one experiment was to determine a feasible method for testing this hypothesis using the DDD-III, to implement the method chosen, and to obtain results for later analysis.

The situations in which common functional commanders were expected to add value formed the basis for the two specific hypotheses tested in this experiment:

- An organization with a common functional commander is better for tasks that require coordination between units in this functional area for the use of assets owned by one of them.

- An organization without a common functional commander is better for tasks that require coordinated use between units in this functional area of assets not owned by any of them.

The specific variables manipulated during the phase one experiment revolved around levels of competition that can were created by limiting assets. Further discussion of these manipulations is addressed by Berigan [Ref 4 and 5].

B. IMPLEMENTATION

The experiment was conducted using a network of SUN SPARC^{TM} workstations located in the Naval Postgraduate School - Systems Technology Laboratory (STL). The players were physically in the same room, although dividers were placed between them to limit any scenario knowledge gained by observing the other team members. Talking was restricted during the simulation runs, with communications among players restricted to the preformatted computer messages built into the DDD-III.

C. SCENARIO DESCRIPTION

1. Geographic/geopolitical

The scenario chosen was representative of a "typical" future requirement for US forces. Orange, a North African nation friendly to the United States, has been attacked by Green, whose

forces have taken control of Orange's port of Eastport. Upon direction from the National Command Authority a Joint Task Force (JTF) has been organized by the theater commander in chief, the Commander in Chief, Mediterranean Command (CINCMED), in order to capture the port and a nearby airfield to allow for the introduction of follow-on forces. The Commander, JTF (CJTF) has at his disposal an aircraft carrier battle group (CVBG) and an amphibious ready group (ARG) that transports two Marine Expeditionary Units (Special Operations Capable) (MEU(SOC)s).

2. Mission and Execution

Green forces are located at the port, the airfield, and at other locations in the amphibious objective area. About 5 miles south of the port, there are two suitable landing beaches with a road leading from the northernmost beach (designated "Red Beach") to the port, and another leading from the southernmost beach (designated "Blue Beach") to the airfield.

The forces from MEU1 and MEU2 will land at Red and Blue Beaches, and proceed along the roads to capture and secure the port and airfield, respectively. They will use their own assets and request the CJTF's assets as necessary to clear mines at the beaches, conduct MEDEVACs, defeat counterattacking armored units, clear mines along the roads, suppress Green artillery, destroy Green FROG launchers, and capture the port and airfield.

The maritime units (CVBG and ARG) support the amphibious operation with CAS, NSFS, mine countermeasures, and air defense assets, while defending themselves against air,

surface, and subsurface threats. They will use their own assets and request the CJTF's assets to destroy Green submarines, fixed-wing aircraft, helicopters, patrol boats, and Silkworm launchers, as necessary.

3.   Competition

As stated earlier, competition is used to "force" coordination among the DMs. The competition for assets is operationalized by manipulating the tasks presented to the decisionmaking team so as to require assets in several places simultaneously. The two major command structures utilized during the scenarios were designed to allow for asset competition at either an organic or a non-organic level. Each scenario module represented asset competition for a different DM team - for example, scenario "A" presented situations that could be resolved if the organic assets were properly coordinated. The ground component units (MEU1 and MEU2) competed for MEU1's engineer platoon and Cobras and MEU2's MEDEVAC helicopters. The maritime component (ARG and CVBG) competed over the ARG's Stinger platoon and the CVBG's Aegis cruiser, frigate, and section of CAP aircraft. Non-organic assets that were not competed for, but were used in the scenario. In scenario "B" the organic and non-organic assets were the same (as in scenario A), however, the scenario was crafted such that organic assets were not competed for, while the non-organic assets were. Scenario B unfolded as did Scenario A, except that competition occurred in different events (i.e., "tasks").

D. ABSTRACTION

The real challenge of implementing this experiment was in the attempt to accurately depict elements of the real-world and the joint environment through the use of the DDD. The higher level simulators examined in chapter two all contain displays with exceptionally high resolution of the tactical picture, which of course requires extensive software. The DDD simulation required that the decisionmakers be able to identify basic geography, threats and resources as an abstraction of the real-world. Additionally, crucial elements of command and control needed to be modeled. A brief description of these various abstracted elements is contained below:

1. Information structure

One of the major assumptions behind the concept of organizational "flattening" [Ref 5] is the existence of a common operational picture (COP). All commanders at all levels must have a common view of the battle space - they must see the same threats, at the same time. Since our purpose was to test organizational structures in a future environment of shared, global information, the COP concept was a "given" for this experiment. When one decisionmaker in the organization saw a threat or task, it was seen by all others at the same time. It was felt that this common view might reduce parochialism in certain circumstances, through fostering of shared mental models among team members.

The DDD-III, however, has the capability to control the amount of "shared" information available to the DMs. This is accomplished using the *Task View* command, which determines the team information structure for the detection and measurement of a specific task class by any

41

platform owned by a specific DM. This command can be used by the simulation designer to tailor the tactical picture of the DMs to model various degrees of "Global awareness".

Example: task view 0 2 2 2 2 2 2

Format: task view id   v(0)   v(1) v(2) ... v(ndm-1)
       # id: integer number, uniquely identifying the task class.
      # v(I):   number which specifies the degree to which the decisionmaker
             Dmi can observe this class of tasks.

      = 0: DMi cannot see a task of this class unless the task happens to be
           within the detection range of at least one of his own sensors.
      = 1: DMi can see a task of this class if the task is detected by any of the
           sensors on platforms owned by other DMS.
      = 2: DMi can see a task of this class and get attribute/class information
           if the task is within the measurement/identification ranges of any
           of the sensors on platforms owned by other DMS.

Figure 4-1: example of task view command

An example of *task view* is shown in Figure 4-1. In this instance, task class 0 task view was set to 2, which allowed all DMs to see a task 0 contact once in range of <u>any</u> platform sensor.

2.   Command structure

Two distinct levels of organizational structure were used for this experiment, each with its own unique requirements and characteristics:

-     Three tiered: Composed of a CJTF plus a common functional commander, either a ground component commander (GCC) or a maritime component commander (MCC), supervising the two lowest-level units (MEU1 and MEU2, or the CVBG and ARG, respectively).

- Two-Tiered, with the lowest-level units (MEU1 and MEU2, or CVBG and ARG) reporting directly to the CJTF

Although the two-tiered and three-tiered structures were separated for analysis, the two JTF organizations that were used for the experiment each had an intermediate commander supervising one component and none supervising the other. Thus, in half of the runs there was a GCC, while in the other half there was an MCC. This was done in order to keep the number of subjects constant across all trials, and avoid task-load-per-individual problems that would have

Figure 4-2: Graphical representation of command structure
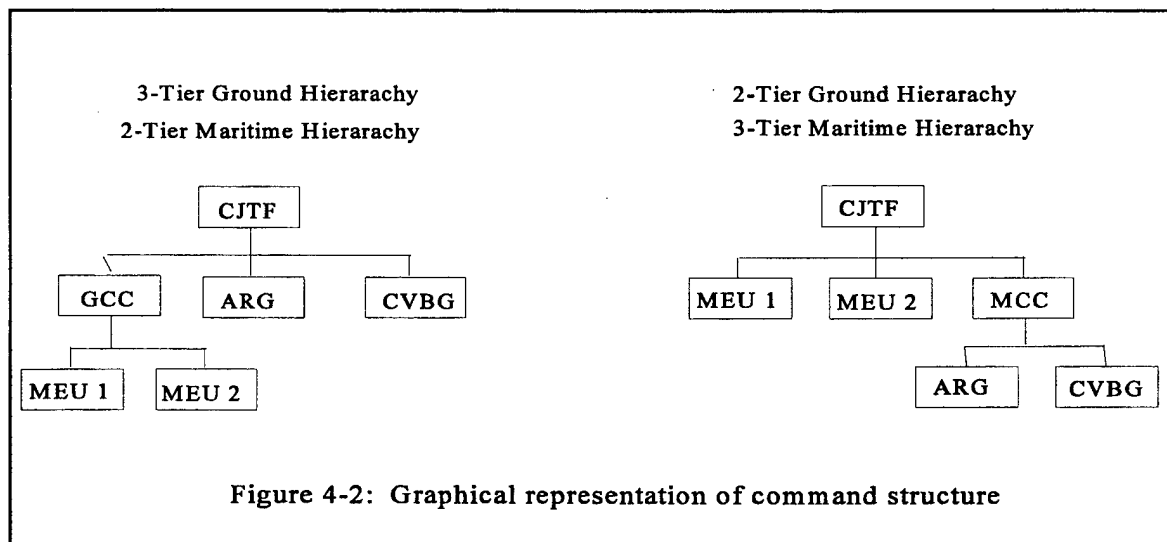
arisen had the two structures been composed of different numbers of subjects. The two organizational structures are depicted in Figure 4-2. The command and communications structures were operationalized via two separate commands.

a.   Decision Structure

The *decision structure* command is used to define the number of decisionmakers present in the simulation, as well as the command structure they are contained within. (in reference to the other decisionmakers) This allows the scenario designer to specify command structure, which affects authority/responsibility and asset control issues among decisionmakers. Only acyclic structures are supported in DDD-III.

```
Example:      decision structure 6
              CJTF   GCC   MEU1   MEU2   CVBG   ARG
              0      0     1      1      0      0

Format:  decision structure ndm
         name(0) name(1) ... name(ndm-1)
         cd(0)  cd(1) ...  cd(ndm-1)
```

Figure 4-3: example of decision structure command
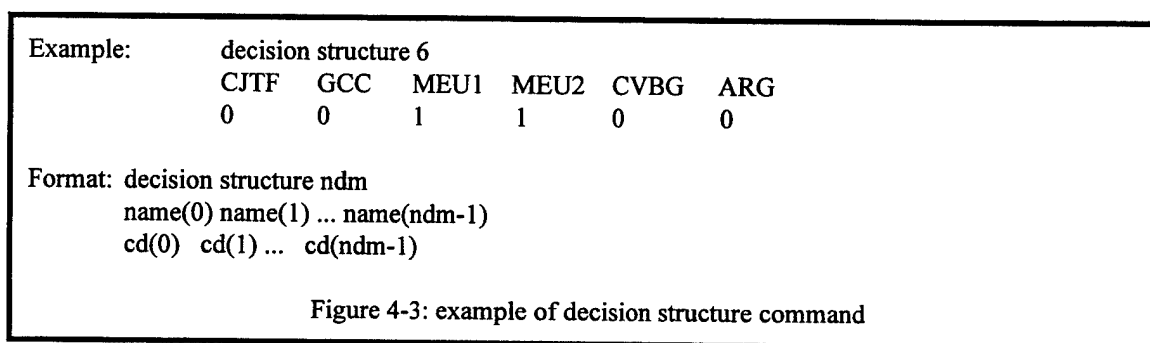
Figure 4-3 shows the decision structure assigned for one set of phase one scenarios. There are six decisionmakers defined for this scenario. DM1 is assigned the name CJTF, DM2 is assigned the name GCC, etc. This command sets the three tiered ground, two tiered marine structure previously discussed, with CJTF as overall commander, GCC, CVBG and ARG reporting to the CJTF and the two MEU's reporting to GCC.

b.   Communications Permission

After the command hierarchy is determined the question of communications connectivity must be addressed. Although the DDD-III provides the ability to address command and communications separately, within this phase one experiment the communications abilities paralleled the command structure. The *communications permission* command sets the

communications structure for the decisionmakers within the simulation. If a communications link exists from decisionmaker i to j, DMi can request/send resources, information and action/intention messages to DMj. The communications structure used for one set of phase one experiment scenarios is shown in Figure 4-4.

```
Example:        communication permission
                0   1   1   1   1   1
                1   0   1   1   1   1
                0   1   0   1   1   1
                0   1   1   0   1   1
                1   1   1   1   0   1
                1   1   1   1   1   0


Format: communication permission
        P(0,0)      P(0,1)   ...  P(0,ndm-1)
        P(1,0)      P(1,1)   ...  P(1,ndm-1)
        P(ndm-1,0)  P(ndm-1,1) ... P(ndm-1, ndm-1)

# P(i,j)  = 1, DMi can communicate with DMj.  (i.e. link exists from I to j)
        #      = 0, DMi can't communicate with DMj.
        # Default: all diagonal elements are 0, the rest are 1
```

Figure 4-4: example of communications permission command

For this example (three tiered ground, two tiered maritime) all decisionmakers could communicate with each other except for MEU1 and MEU2, who were forced to pass all requests for CJTF's assets and messages through the GCC. This was implemented by setting all communications between DMs to 1, with the exception of the MEUs to CJTF connections, which were set to 0, preventing direct communications.

3.   Own Forces available

Assets owned by the five organizational entities (the sixth, the GCC/MCC depending on the scenario, did not directly own assets) were derived from U.S. assets currently available in the

real-world. Each asset had to be represented within the scenario as a platform. Those assets chosen and available for use in this scenario are as follows along with their <u>initial</u> owner:

a.   *MEU (SOC) 1*

One AAAV-mounted infantry company

One V-22 Osprey-mounted heliborne infantry company

One division (4) AH-1W Cobra attack helicopters (indivisible)

One V-22 mounted combat engineer platoon

Naval Surface Fire Support (NSFS) from one destroyer

b.   *MEU (SOC) 2*

One AAAV-mounted infantry company

One V-22 Osprey-mounted heliborne infantry company

One section (2) MEDEVAC helicopters (indivisible)

NSFS from a second destroyer

c.   *CVBG*

Aircraft Carrier

AEGIS cruiser

Antisubmarine Warfare (ASW)-capable frigate

Combat Air Patrol (CAP) section

d. *ARG*

Two destroyers providing NSFS for MEU's

Amphibious ships

One V-22 mounted Stinger platoon (indivisible)

One section of CAP

e. *CJTF*

Two sections of close air support (CAS) aircraft

One mine countermeasures (MCM) helicopter

One section of SH-60 helicopters for anti-surface warfare (ASUW)

One V-22 mounted heliborne infantry company (JTF reserve)

SR-71 photo reconnaissance mission

One section of JFACC F-15's for air defense

The scenarios were designed so that if a DM must perform a specific task, then the assets

required should be transferred to that DM, rather than the original owner attempting to perform

the task for the unit that required the asset. For example, if MEU2 was under attack by a tank

column, we wanted MEU1 to transfer the necessary asset (the Cobra helicopters) to MEU2 to

destroy the tanks, rather than MEU1 trying to destroy the tanks itself.

The abstraction of the assets into DDD-III platforms required that the function of the real-

world platform be reviewed and that the value of the resources be assigned accordingly. The

resource vector was chosen to contain seven individual elements, corresponding to the asset's ability to perform the following seven missions:

- Air - attack air targets
- Sea - attack sea target
- Grnd - Attack ground targets
- Hold - once taken, the ability to hold ground
- Mine - the ability to sweep for mines, both land and sea
- Armor - the ability to attack armor
- Med - the ability to conduct medivac missions

A listing of the platform classes and their resources is contained in Table 4-1. These resources reflected the relative "power" assigned to that platform class based on real-world capabilities of assets. For example - a land based fighter (class 2) has tremendous capability against air threats, but is less effective against sea and ground threats. The fighter has no ability to hold territory, destroy mines or conduct a medivac mission, so the value for these attributes is zero. All assets in the scenario were transferrable, with few exceptions. The non-transferrable assets included such assets as the amphibious shipping and the aircraft carrier, which were not directly needed to accomplish tasks.

If the asset owner chose not to transfer an asset as requested by a DM needing it, a higher-level decisionmaker could force transfer of the asset(s). For example, if the ARG requested an asset from the CVBG, and the CVBG ignored the request, the MCC (if present) or CJTF could

forcibly transfer the asset from the CVBG to the ARG, if he determined that the ARG's need for the asset outweighed that of the CVBG.

Assigned Resources Values

# Platform name	Air	Sea	Grnd	Hold	Mine	Armor	Med
0 Test platform	50	50	50	50	50	50	50
1 Carrier Fighter	5	2	0	0	0	0	0
2 Land based fighter	5	1	1	0	0	0	0
3 Carrier strike	0	5	5	0	0	5	0
4 MCM Helicopter	0	0	0	0	5	0	0
5 ASUW helicopter	0	6	0	0	0	0	0
6 Cobra helicopter	0	0	5	0	0	10	0
7 Huey helicopter	0	0	5	5	0	0	0
8 Huey (medivac)	0	0	0	0	0	0	5
9 engineering platoon	0	0	0	0	5	0	0
10 Recon aircraft	0	0	0	0	0	0	0
11 Standard missile	5	1	0	0	0	0	0
12 NSFS mission	1	1	5	0	0	2	0
13 Stinger Det	6	0	0	0	0	0	0
14 FFG (ASW platform)	1	5	0	0	1	0	0
15 Carrier	1	1	0	0	1	0	0
16 CG (AAW platform)	1	1	0	0	1	0	0
17 DDG (NSFS platform)	1	1	0	0	1	0	0
18 DDG (NSFS platform)	1	1	0	0	1	0	0
19 Amphibious platform	1	1	0	0	1	0	0
20 Amphibious platform	1	1	0	0	1	0	0
21 Amphibious platform	1	1	0	0	1	0	0
22 LCAC 1	0	0	0	0	0	0	0
23 LCAC 2	0	0	0	0	0	0	0
24 AAAV	0	0	5	5	0	1	0
25 Shore bases	0	0	0	0	0	0	0

Table 4-1: Assigned values of Platform resource vectors

f.   *Subplatform structure*

Some of the platforms available to the DMs were represented as subplatforms, representing real-world asset structure. Figure 4-5 shows the assignment of subplatforms to platform used in this phase experiment. The number of subplatforms available to the DM was a combination of real-world capabilities and the desire to drive competition for the assets. A more detailed description of the *platform Subplatform* command can be found in chapter five.

49

```
#Aircraft carrier carrying: fighters, Combat Air Support(CAS), anti-ship helicopters
 platform Subplatform  15 3
      VF   VA    H60
      3     2     1
      4     0     0
#
# Cruiser carrying: SAM missiles
platform Subplatform  16  1
      SAM
      10
      -1
#
# Destroyer 1 (MEU 1 asset) carrying 5 Inch fire mission
platform Subplatform  17   1
      5I
      10
      2
#
# Destroyer 2 (MEU 2 asset) carrying 5 Inch fire mission
platform Subplatform  18   1
      5I
      10
      3
#
# amphibious ship (MEU1) carrying: AAAV and attack/troop/engineer helicopter
platform Subplatform  19  3
      HCB  HTP  HE
       1     1    1
       2     2    2
#
# amphibious ship (MEU2) carrying: troop/medivac/engineer helocopter
platform Subplatform  20  3
      HTP   SD   HMV
       1     1     1
       3     5     3
#
# amphibious ship (CJTF) carrying : mine-countermeasure helicopter/troop helicopters
platform Subplatform  21 2
      MCM  HTP
       1      1
       0      0
#
# sigonella base providing: Fighters, reconnaissance
platform Subplatform  25   2
      F15  SR7
       1   1
       0   0
```

Figure 4-5: example of Subplatform assignment for the scenario

4. Enemy forces presented

The scenario was designed so that each threat (or "task" to be accomplished) could be defeated (or accomplished) by only one asset in that given situation. This "matching" of assets to tasks was done in order to induce the proper competition, and internal coordination, among the DMs for limited JTF assets. The DMs were briefed prior to the game and understood the relationship between various threats and the assets needed to properly engage that threat.

a. Threats presented to the landing forces

Mine fields offshore of one or both beaches

One or more company-sized armored units

Mine fields on roads (swamps restrict off-road travel)

A number of artillery strong points

Possible hidden FROG surface-to-surface missile launchers

Heavy mortar platoon in range of Red Beach and port

Infantry units defending the port and airfield

b. Threats to the maritime forces

One or more Alfa-class submarines

MI-24 Hind and fixed wing aircraft, capable of anti-ship actions

Fast patrol boats

One or more hidden Silkworm anti-ship missile launchers

51

The abstraction of the threat into DDD-III tasks required that the function of the real-world platforms be reviewed and that the value of the task attributes be assigned accordingly. These attributes reflect the difficulty of the task as well as the relative importance. Through the mapping discussed in section 5 of this chapter, the attributes were coupled to the resources required. For our scenario the assigned attributes for twenty-three task classes are shown in Table 4-2.

# Task name	Value	Time	Air	Sea	Grnd	Hold	Mine	Armor	Med	Enemy
0 Hills		10.0	20.0	0.0	0.0	5.0	0.0	0.0	0.0	1.0
1 Airport	30.0	20.0	0.0	0.0	10.0	0.0	0.0	0.0	0.0	1.0
2 Seaport	30.0	20.0	0.0	0.0	10.0	10.0	0.0	0.0	0.0	1.0
3 Hold ground	0.0	10.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	1.0
4 Take ground	10.0	10.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	1.0
5 Artillery	2.0	10.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	1.0
6 Frog Launcher	10.0	20.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	1.0
7 Silkworm	15.0	20.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	1.0
8 mines (land)	5.0	10.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	1.0
9 mines (sea)	5.0	10.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	1.0
10 Strike air	15.0	20.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
11 TAC air	4.0	20.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
12 Helicopter	15.0	10.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0
13 neutral air	10.0	10.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14 Tanks	5.0	10.0	0.0	0.0	0.0	0.0	0.0	10.0	0.0	1.0
15 Neutral (grnd)	10.0	10.0	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0
16 Patrol boat	15.0	10.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	1.0
17 Submarine	15.0	10.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	1.0
18 ASCM	15.0	10.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	1.0
19 Neutral sea	10.0	10.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0
20 Medivac	5.0	60.0	0.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0
21 Swamp	2.0	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
22 Silkworm (air)	15.0	20.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	1.0

Table 4-2: Assigned values of task attribute vectors

5. Mapping of Resources (own forces) to Attributes (enemy forces)

A basic element of the experiment was competition between DMs over assets necessary for completing a given task or series of tasks. A mechanism was required to couple the resources

required to prosecute a task with that tasks' attributes. Based on the expected paths taken in the scenario, the platforms provided and the threats presented, the attribute and resource vectors were tailored to force specific platform/threat match-ups. Although the DDD-III allows any asset to attack any threat, as long as the appropriate flags are set, full points are only awarded when the threat is attacked by assets having the correct match-up of resources to those required.

This "mapping" of threat attributes to resources required was accomplished using the task mapping command. This command specifies a linear mapping from the attributes of a task class to the resource requirements for a successful attack on that class. Figure 4-6 provides an example of the command.

```
Example:      task mapping 0
              0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00   0.00
              0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00   0.00
              0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00   0.00
              0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00   0.00
              0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00   0.00
              0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00   0.00
              0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00   1.00

Format:               task mapping id
              A(1,1)    A(1,2)  ...  A(1,natt)   B(1)
              A(2,1)    A(2,2)  ...  A(2,natt)   B(2)
              A(nres,1) A(nres,2) ... A(nres,natt) B(nres)
```

Figure 4-6: Example of task mapping command

This structure of the mapping for the resource vector allows for an assignment of attribute values to resource values and for the addition of a constant. This allows the simulation designer to control the resource match up between tasks and platforms, which enables a representation of

53

the differences in firepower and engagibility found in the real-world. The specific assignment of attributes to resources for the scenario is shown in Figure 4-7.

Attributes		Resources	
#	Name	#	Name
1	Value		
2	Time		
3	Air	1	Air
4	Sea	2	Sea
5	Ground	3	Ground
6	Hold	4	Hold
7	Mine	5	Mine
8	Heavy/armor Task	6	Heavy/armor
9	Medivac	7	Medical
10	Enemy (Recon)		

Figure 4-7: Attribute to Resource assignment

For example, the VA, or strike aircraft platform, was the only asset with the correct resource values to attack the frog launcher, thus causing possible competition between the DMs over that asset. This example is shown in Figure 4-8. The armor resource of the strike aircraft was set to 5.0, which matched the armor attribute of the Frog launcher. Because the mapping of the armor resource to attribute was one-to-one, any attack on the frog launcher by a VA aircraft would be considered successful.

	Value	Time	Air	Sea	Grnd	Hold	Mine	Armor	Med	Enemy
Frog Launcher	10.0	20.0	0.0	0.0	0.0	0.0	0.0	5.0	0.0	1.0

	Air	Sea	Grnd	Hold	Mine	Armor	Med
VA (strike aircraft)	0.0	5.0	5.0	0.0	0.0	5.0	0.0

Figure 4-8: Resource and attribute vector comparison

V. DDD-III EXPERIMENT PHASE I - SIMULATION DEVELOPMENT ISSUES

As in most computer simulations the challenge to the simulation designer is to model the "real", physical world within the software world. From an experimental viewpoint it is crucial that the model operate according to real-world rules. A major artificiality can cause unwanted deviations in the results. It is equally important that the displays mimic as closely as possible the information provided in the real-world. Many unique applications of the generic DDD-III commands were developed in order to represent the Joint world as accurately as possible. Elements of the real-world needed to be abstracted that did not fit the generic mold of a direct threat - such as geographic features and stand-off weapons. The challenge was to model these elements without extensive software modifications or complex commands that would consume the scenario designer's time and distract from the true goal of the simulation.

A. GEOGRAPHY

Since nearly all simulation information is provided to the DMs via a monitor display, it is important from a playing point of view that the display be both understandable and reasonable. For the results of the experiment to be valid it is also important that the display mimic the information provided in the real-world. Some elements of the display were easy to model - land and sea areas were created with little modification to the existing software. Others, however, were fairly complex to represent

and required work around solutions to be developed. Several of the more important geographic representations are covered below.

1.   Beachhead

The scenario's general premise, that an amphibious landing would be the primary method of putting troops on the ground, required that a beachhead landing area be provided to the DMs. Additionally, the designers needed to "force" the location of the landing areas to set up follow-on engagements, (i.e. a starting point). From a practical point of view, the beach areas needed to be clearly defined so that the MEU commanders had an idea of where they could place elements ashore. Finally, we knew that during the scenario elements of the MEU would remain at the beachhead, in order to perform medivac tasks; the enemy would target these elements, using the beaches as target areas for artillery attacks.

We chose to implement the beach areas as penetration zones, which clearly indicated to the DMs where the beaches were located and allowed a method for monitoring and scoring of attacks by enemy forces. Via the *task penetration* the beach penetration zone(s) were set to be sensitive to only artillery and Frog missile task classes, which represented forces used by the enemy against the landing area. At various preset times during the scenario an artillery threat would appear, which if not suppressed would (after a set period of time) attack the beach area. The penetration zones and sensitivities are shown in Figure 5-1.

```
#---------
penetration number 7
#---------
penetration zone  circle    0 25.00  45.00  5.00      ( port area)
penetration zone  rectangle 2 26.5   70.00  4.00  6.0 ( beach area one)
penetration zone  rectangle 3 26.5   80.00  4.00  6.0 ( beach area two)
penetration zone  circle    4 5.00   95.00  5.00      ( Airfield)
penetration zone  circle    5 70.00  15.00  7.00      ( fleet area (CVN))
penetration zone  circle    6 64.00  75.00  4.50      ( fleet area amphib))
#
#---------
task penetration 0  0 0 0 0 0 0 0 (hill mission)
  .          .        . .. . . ...  .
  .          .        . .. . . ...  .
task penetration 5  0 0 1 1 0 0 0 (artillery)
task penetration 6  0 0 1 1 0 0 0 (frog missile)
  .          .        . .. . . ...  .
task penetration 23 0 0 0 0 0 0 0 (silkworm - launcher)
#---------
```

Figure 5-1: Example of penetration zone command with task penetration defined

Other areas also implemented as penetration areas were the port and airfield, as well as the two fleet areas located around the CVBG and the ARG. This provided an easy method to target friendly forces with various enemy threats, such as the silkworm missiles and tactical aircraft.

2.   Cities

In the earlier DDD-II paradigm there were no enemy "bases". The simulation only allowed the DMs to defend areas, not to go on the offensive. This was not acceptable in the DDD-III paradigm. We needed to abstract the mission objectives of a typical Joint military scenario, which included offensive or strike capability or movement.

The scenarios in the experiment included several cities, an airport and a port facility, all of which would be important to the amphibious landing forces. The cities were relatively passive representations - the only requirement was that they exist. The scenario required that the DMs discriminate between real and "false" silkworm launchers within the cities to complicate the strike issue (i.e., minimize civilian casualties).

```
#---------
draw rectangle 27.0 49.5 2.5 4.0
draw rectangle 27.0 64.5 2.5 4.0
draw rectangle 27.0 87.0 2.5 4.0
#---------
```

Figure 5-2: Example "cities" using draw rectangle command

As shown above in Figure 5-2, the cities were displayed as small rectangles, their locations included in the OPORDER brief presented to the decisionmaking team prior to the start of the scenario.

3. Port/Airport areas

The airport and seaport required a more extensive representation than did the cities. Both the city and the port were final mission objectives, so the implementation method needed to not only provide a visual reference for the DMs but also to specify the resources required to take the mission objective. The port and city needed to be set up as individual tasks to be accomplished, thus providing a mechanism to record team mission completion. The solution was to create a separate task class for each objective.

```
# task 1: ground mission (airport)
task general 1  G  AP  0.00  0.211  9 dullas.icon
task mean  1  30.0  10.0  0.00  0.00  10.00  10.00  0.00  0.00 0.00  1.0
task view  1 2  2  2  2  2  2
task stealth 1  0 100.0 100.0 100.0 00.0 0.0
   .        .    .   .    .       .      .    .   .
   .        .    .   .    .       .      .    .   .
task stealth 1 23 100.0 100.0 100.0 00.0 0.0
task attack  1  1  1 1 1 1 1 1
#
# task 2: ground mission (seaport)
task general 2  G  SP  0.00  0.211  9 norfolk.icon
task mean  2  30.0  10.0  0.00  0.00  10.00  10.00  0.00  0.00  0.00 1.00
task view  2 2  2  2  2  2  2
task stealth 2  0 100.0 100.0 100.0 00.0 0.0
   .        .    .   .    .       .      .    .   .
   .        .    .   .    .       .      .    .   .
task stealth 2 23 100.0 100.0 100.0 00.0 0.0
task attack  1  1  1 1 1 1 1 1
#---------
```

Figure 5-3: Example of mission tasks (Airport and Seaport)

Figure 5-3 shows both the airport and seaport task classes created. The task

classes created had no velocity and were in a fixed location. The threat flag assigned was

a 9, which indicated to the scenario generator that both tasks represented missions to be

accomplished. Unique icons were used for both the airport and the seaport. The task

view command, as well as the task stealth command, were set so that the city and port

were known and visible to all DMs throughout the entire scenario, since the location of

major cities would be known to the forces landing.

Each task class required a unique combination of ground and heliborne assets to

properly complete the task, which was done by using the *task mean* command to assign

unique values for the air and ground attributes. The values assigned assured that the two

59

ground force DMs, MEU1 and MEU2, would require additional assets to complete their missions, creating possible competition between the DMs.

4. Land obstacles

We wanted to closely model the real world environment encountered by troops, which would contain roads, impassible areas, hills, etc. Additionally we had an experimental requirement - the two ground forces needed to advance towards their objectives, and reach them, at roughly the same time. A mechanism was needed to prevent a "mad dash" to the objective by the ground forces. There obviously would not be any competition of assets if the forces were so out of sync that one could use the needed asset, while the other force was still on the beach. This "slowdown" of the forces was accomplished as follows:

a. Roads

The primary mechanisms employed to "channel" the ground forces were roads, connecting the beaches to both the airport and the seaport. The teams were briefed, via OPORDER, that off-road travel was prohibited and that the ground was to be considered impassable. A line drawing command, which gave the designer the ability to draw lines between sets of XY coordinates, was added to the DDD-III to provide the ability to represent the roads in the scenario.

b.   Swamps

After the roads were clearly defined the question still remained how to

ensure the ground units would travel only on the roads. Though there was a chance that

the teams would choose to remain on the road without incentive, it was felt that a

mechanism was needed to make the land truly impassable, ensuring compliance with the

OPORDER. The general philosophy used was to make leaving the road a bad choice,

but not an asset killer or game ender. A penalty (loss) would be assigned to the team,

regardless of which DM caused the violation.   The penalty was large enough to be

annoying but small enough that overall performance measures would not be skewed.

Implementing this impassability condition required tailoring a task class to represent a

"swamp" threat, shown in Figure 5-4.

```
# task 21: Swamp
task general 21  G  SM .00  0.211  2 swamp.icon.no_label
task mean  21  2.0   10.0  0.00 0.00   20.00  0.00   0.00  0.00   0.00 0.00
task view  21  2  2  2 2 2 2
task stealth  21  0   100.0 100.0 100.0 00.0 0.0
  .                .   .       .        .       .     .   .
  .               .  .       .       .          .    .  .
task stealth  21 21  100.0 100.0 100.0 00.0 0.0
task stealth  21 24  100.0 100.0 100.0 00.0 2.0
task stealth  21 25  100.0 100.0 100.0 00.0 0.0
task attack   21  1  1 1 1 1 1 1
#----------
```

Figure 5-4: Example of swamp task class definition

Task class 21, the "swamp", was of type G (ground) and had 0 velocity. Using

the *stealth* command (discussed further in this chapter) the swamp task was assigned a

be-attack radius of 2 miles for ground units. This allowed us to spatially distribute the

tasks, creating an area of impassability without completely covering the display with swamp icons. The assigned value for the penalty of contacting a swamp icon was 2.0 - very small compared to other task values. The swamps had an attack radius of 0 miles - the task could not be attacked by any platform. This effectively prevented the ground units from "clearing" their way across the land area by destroying the swamps. (Which is not an option in the real world).

In the real world swamps would be a known quantity to a ground force <u>prior</u> to landing. To ensure that the swamps location <u>was</u> known to the DM's <u>prior</u> to the amphibious landing, the detection and identification ranges for all platforms was set to 100 miles for the swamp task class. This resulted in all DM's, even if not <u>directly</u> involved in the ground action, knowing the location of the swamp areas. Full knowledge of the <u>exact</u> geographic features in a foreign country, by a landing force, is a bit optimistic, but for this simulation the artificiality can be overlooked. The swamp's most important function was to "channel" the landing forces into a specific route, guaranteeing a minimum time to reach the objectives. The swamp coverage was sufficient and the penalty severe enough that the ground units did in fact remain on the roads, a result of which was that the pretimed mission events went well.

The combination of roads, cities, seaport, port and swamp icons located in the land area made for a rather full, but not overly cluttered display. Once briefed in regards to the land area representation, using the tutorial contained in Appendix B, the simulation

subjects acting as DMs had no trouble negotiating the represented geographic features.

An example of the land area display is contained in Figure 5-5.
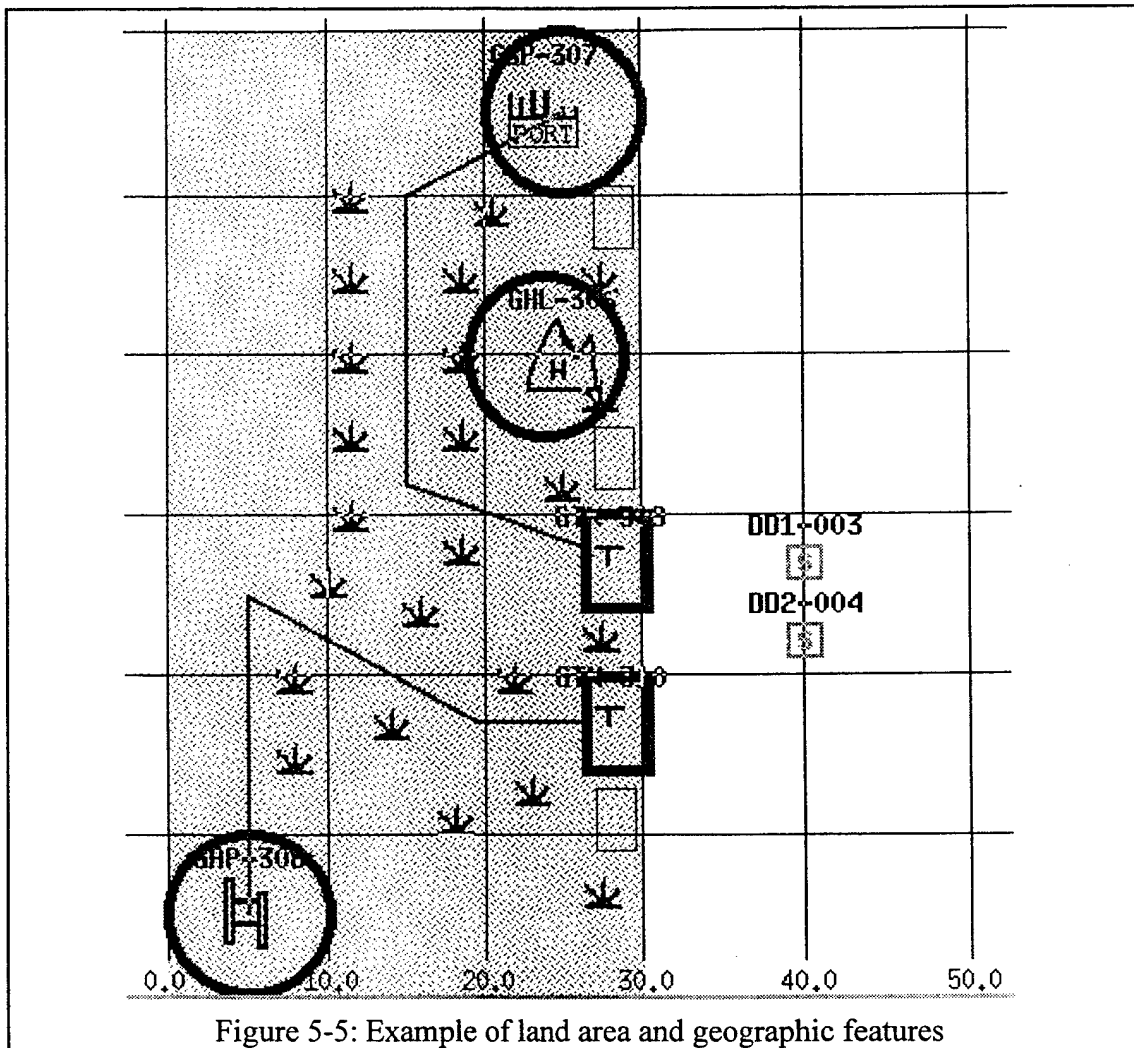
Figure 5-5: Example of land area and geographic features

B. PLATFORMS

Numerous issues relating to platforms developed during the planning stage for the phase one experiment. Most platforms were fairly easy to represent using the DDD - assets such as fighters, tanks and ships had a "known" capability that did not change

63

much from platform to platform. Platform design followed a fairly set pattern. The scenarios generally called for assets based on what is currently available to the CJTF in a typical regional conflict. There were, however, several platforms that required additional development.

1. Reconnaissance platform

In order to force ground elements to compete for non-organic assets we placed all reconnaissance assets under the ownership of the CJTF. An SR71 reconnaissance asset was the only asset capable of determining whether a silkworm missile site was a real site or a decoy. Based on this information, gathered by the SR7 platform, a decision would be made by the DM to attack or ignore the threat. The reconnaissance platform command is shown in Figure 5-6.

```
# platform 10: recon aircraft
platform general 10  A   SR7  0.50  1

platform resource 10   0 0 0 0 0 0 0
platform range   10
 25.00  20.00  15.00   0.00  0.01
 50.00  50.00  45.00   0.00  0.01
 50.00  50.00  45.00   0.00  0.01
platform accuracy  10
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 *0.000*
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 *0.000*
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 *0.000*
#
```

Figure 5-6: Example of the SR7 Platform Class

To implement this identification ability the SR71 task class needed a unique capability - the ability to measure the value of the "enemy" attribute, a_{10}, contained in the task attribute vector. The SR7 platform was given this ability using the *platform accuracy* command. The SR7 was given the ability to "see" the attribute - all other platforms were unable to measure this value of the attribute vector, as their platform accuracy on attribute a_{10} was set to -1, which prevented measurement of that attribute.

2. Platform reusability

The issue of whether a platform could be used more than once became important when trying to set a time line for task accomplishment by the team. The ability to induce competition within the team depended on the number of assets available at any given time. Two mechanisms were used to control the assets available:

a. Endurance/returnability

The endurance of a (sub)platform is set in the *platform general* command. This command specifies how long a subplatform exists once it has been launched. The endurance specification is germane to the scenario generator only in the case of a subplatform, but since the scenario generator allows <u>any</u> platform to be designated as a subplatform the endurance must be defined for all platform classes. If the platform class is not designated as a subplatform the endurance of the asset would be for the entire scenario period by default. Some assets used as subplatforms, such as fighters and attack

65

aircraft, had a limited endurance period after which they would return to their point of origin (i.e. carrier, base, etc.)

Whether the asset could be launched again was determined by the returnability flag, also set in the *platform general* command. The intent here was to model cycle times such as in aircraft operations, which in the real-world limits the air assets available at any given time.

b. Reusable flag

The life span of "consumable assets", such as fire support rounds and missiles, was too hard to predict with any accuracy, so the endurance value was not much good - another method to limit use was needed. The reusable flag, also set in the *platform general* command, provided for "single use" subplatforms to be made available. If the subplatform was used to attack a task this flag determined whether or not the asset would still be available after the attack was complete. The reusable flag was most useful with the "consumable" assets, such as surface to air missiles or naval surface fire support rounds. One "consumable" asset used in the scenario was Naval Surface Fire Support (NSFS). This subplatform represented a "fire mission", vice a single round from a gun, but the concept of reuse still applied. When requested from the supporting surface ship, the fire mission subplatform was available for a limited time. Once used, the number of NSFS missions available to the MEU decreased by one - limiting the number of missions available to the DM, much like the real-world. An example of reusable flag use is shown

in 5-7. The *reusable flag* in the *platform general* command is set to zero, indicating the platform was a single use "consumable" asset. This resulted in the 5I platform disappearing along with the destroyed task. Amplification of the *platform general* command can be found in Appendix A.

```
    # platform 12: ship based NGFS support rounds
    platform general 12  A   5I  0.00  0
    0   0   60.00  3.000        10.000
    platform resource 12   1  1  5  0  0  2  0
    platform range   12
    0.00  0.00  0.00  0.00  0.00
    0.00  0.00  0.00  0.00  0.00
    45.00  0.00  0.00 40.00  0.00
    platform accuracy   12
    0.000  0.000  0.000  0.000  0.000  0.000   0.000  0.000  0.000  -1.000
    0.000  0.000  0.000  0.000  0.000  0.000   0.000  0.000  0.000  -1.000
    0.000  0.000  0.000  0.000  0.000  0.000   0.000  0.000  0.000  -1.000
```

Figure 5-7: Example of the NSFS platform used as a subplatform

3.   Subplatform ownership - location vs command hierarchy

Subplatform ownership became a critical issue during the implementation phase of the scenario design. The scenario called for subplatforms to reside on a platform until needed, then to be launched by the platform owner. The initial version of the DDD-III automatically assigned ownership of the subplatform to the DM who owned the parent platform. This made the abstraction of theater assets extremely difficult. There was no way to represent subplatforms, owned by one DM, that resided on a platform owned by another DM, such as theater assets stationed on the carrier. The command *platform subplatform* was developed to allow full flexibility in regards to subplatform ownership.

This command specifies the number of subplatform classes that a specific platform class could have, as well as which subplatform classes (by name), and how many were contained on each platform. Additionally, and perhaps most importantly, the command specifies the ownership of the subplatform following launch, allowing the designer to place assets on platforms but retaining ownership with a second DM. The structure of this command is shown in Figure 5-8.

```
Format: platform subplatform id  num
        sub_class(1)     sub_class(2) ...   sub_class(num)
        n(1)             n(2)              ...    n(num)
        owner(1)         owner(2)          ...   owner(num)
# id:    integer, giving the unique identification of the parent platform class.
# num: integer, indicates the number of subplatform classes that this platform class has, max=7

 NOTE: The maximum number of subplatforms that can be 'on-screen' at any one time is 100.

# sub_class(i):   the name of the subplatform class.  This name must be one of the class given in an
                  earlier platform general command.

 NOTE: The platform and subplatform classes must be defined (via the platform general
command)
        BEFORE the definition of their nesting structure.

# n(i):  the number of individual subplatforms of class 'sub_class(i)' on board this platform class.
# owner(i): the number of the decisionmaker who will own the individual subplatforms of class
           'sub_class(i)' when they are launched.

 NOTE: If owner(i)= -1 these subplatforms will be owned by whomever "owns" the parent
platform
        e.g. this gives the designer the ability to model self defense weapons.

 NOTE: The default is no subplatforms present on platforms

                Figure 5-8: Example of platform subplatform nesting structure
```

```
Example:              platform subplatform 15  3
                      VF      VA      H60
                      3       2       1
                      4       0       0
```

Figure 5-9: Example of platform subplatform command use

In the example of *platform subplatform* shown in Figure 5-9, each parent platform

of platform class 15 (an aircraft carrier) will have three subplatform classes (children)

onboard. The three subplatform classes are VF(fighter aircraft), quantity 3, owned by

DM4 (the CVBG commander); VA (strike aircraft), quantity 2, owned by DM0 (the

CJTF commander); and H60 (helicopters), quantity 1, also owned by DM0.

C. TASKS

1.   Specific task issues ("indirect" threats)

Most threats were relatively easy to model - they entered the tactical picture,

maneuvered through the region towards a target objective (most often a penetration zone)

and attacked, if not stopped by the team. To accurately model the real-world, however,

several classes of tasks were needed to represent the so-called indirect threats - contacts

that appeared in one area but were actually a threat to friendly forces in another area.

Examples of this type of threat are artillery and silkworm missiles.

a.   Artillery

The scenario required enemy artillery units to appear within the general vicinity of the ground forces and target friendly positions. If left unmolested the artillery would eventually fire on the beachhead landing areas. An objective of the JTF organization was to suppress the artillery positions prior to actual firing, protecting any assets on the beaches. An example of the task class artillery is shown in Figure 5-10.

```
#          .
# task 5: ground contact (artillery)
task general 5  G  AT  .99 0.211  2  artillery.icon
task mean  5  2.0  10.0  0.00  0.00  0.00  0.00  0.00  2.00  0.00 1.00
task view  5 2  2  2  2  2  2
task stealth  5  0  100.0 100.0 100.0 100.0 00.0
task stealth  5  1    00.0   00.0   00.0    5.0 00.0
task stealth  5  2    00.0   00.0   00.0    5.0 00.0
task stealth  5  3    00.0   00.0   00.0    8.0 00.0
task stealth  5  4    00.0   00.0   00.0   00.0 00.0
task stealth  5  5    00.0   00.0   00.0   00.0 00.0
task stealth  5  6    00.0   00.0   00.0    6.0 00.0
task stealth  5  7    00.0   00.0   00.0    5.0 00.0
task stealth  5  8    00.0   00.0   00.0   00.0 00.0
task stealth  5  9    00.0   00.0   00.0   00.0 00.0
task stealth  5 10 100.0   50.0   50.0   00.0 00.0
task stealth  5 11   00.0   00.0   00.0   00.0 00.0
task stealth  5 12   50.0   50.0   50.0   50.0 00.0
task stealth  5 13   00.0   00.0   00.0   00.0 00.0

  .      .    . .     .      .       .      .     .
  .      .    . .     .      .       .      .     .
task stealth  5 23  00.0  00.0  00.0  00.0 00.0
task stealth  5 24  50.0  50.0  50.0   5.0 00.0
task attack  5  1  1 1 1 1 1 1
```

Figure 5-10: Examples of artillery task class

To allow for the ground units to see the artillery task a *task stealth* command was used to modify the ground ranges for the MEU ground forces (Task class 24, the AAAV, is

modified with the Task stealth 5 24 line above). The ranges of the 5I platform and the DDG offshore were also modified to allow for engagement (task stealth 5 10 and 5 12).

The team was constrained to the use of a single particular asset, Naval Fire Support (NFS) to suppress the artillery threat. This was accomplished by exclusively mapping the attribute vector of the artillery task class to the resource vector of the NFS platform class (5In). The 5I platform is shown in example 5-11.

```
# platform 12: ship based NGFS support rounds
platform general 12  A   5I  0.00  0
            0 0 60.00  3.000  10.000
platform resource 12   1  1  5  0  0  2  0
platform range  12
 0.00  0.00  0.00  0.00  0.00
 0.00  0.00  0.00  0.00  0.00
45.00  0.00  0.00 40.00  0.00
platform accuracy  12
0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000 -1.000
0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000 -1.000
0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000  0.000 -1.000
#
```

Figure 5-11: Example of 5I platform class

In order to provide enough time for engaging the artillery threat the maneuver commands were spaced in time to present the threat throughout the game. Each artillery task icon would appear and wait for 300 seconds before moving at a high velocity to "hit" either the north or south beach. Artillery tasks were spaced in time to arrive every 400 seconds for the north beach, every 300 seconds for the south beach. The specific maneuver commands for a series of eight artillery attacks on the north and south beaches are shown in Figure 5-12.

71

```
#-------------------------------
# Tasks 241-250 (artillery)
# Artillery positions:        site one: (12.0  67.0 - north)
#                             site two: (15.0  88.0 - south)
#
maneuver definition  241  500.0   |      maneuver definition  246  600.0
s  12.0  67.0  300.0              |      s  15.0  88.0  300.0
m  12.0  67.0  0.99               |      m  15.0  88.0  0.99
e  27.5  82.5  0.0                |      e  27.5  72.5  0.0
#                                 |
maneuver definition  242  900.0   |      maneuver definition  247  930.0
s  12.0  67.0  300.0              |      s  15.0  88.0  300.0
m  12.0  67.0  0.99               |      m  15.0  88.0  0.99
e  27.5  82.5  0.0                |   .  e  27.5  72.5  0.0
#                                 |
maneuver definition  243  1300.0  |      maneuver definition  248  1230.0
s  12.0  67.0  300.0              |      s  15.0  88.0  300.0
m  12.0  67.0  0.99               |      m  15.0  88.0  0.99
e  27.5  82.5  0.0                |      e  27.5  72.5  0.0
#                                 |
maneuver definition  244  1700.0  |      maneuver definition  249  1530.0
s  12.0  67.0  300.0              |      s  15.0  88.0  300.0
m  12.0  67.0  0.99               |      m  15.0  88.0  0.99
e  27.5  82.5  0.0                |      e  27.5  72.5  0.0
#                                 |
```

Figure 5-12: Example of artillery task class maneuvering commands

b.   *Silkworm sites*

The scenario had a need for a threat that would require CJTF

reconnaissance assets (or a national asset) to provide positive identification prior to

prosecution. A Silkworm missile threat was chosen to fill this requirement. The

description of the ground task class silkworm is shown in Figure 5-13.

72

```
# task 7: ground contact (silkworm anti-ship missile battery)
task general 7  G  SWG  0.99  0.211  1  silkworm.icon
task mean  7  15.0  20.0  0.00  0.00  0.00  0.00  0.00  5.00  0.00  1.00
task view   7 2 2 2  2 2 2
task stealth  7  0  100.0 100.0 100.0  0.0  0.0
task stealth  7  1    0.0    0.0    0.0  5.0  0.0
task stealth  7  2    0.0    0.0    0.0  5.0  0.0
task stealth  7  3    0.0    0.0    0.0  8.0  0.0
task stealth  7  4    0.0    0.0    0.0  0.0  0.0
task stealth  7  5    0.0    0.0    0.0  0.0  0.0
task stealth  7  6    0.0    0.0    0.0  6.0  0.0
task stealth  7  7    0.0    0.0    0.0  5.0  0.0
task stealth  7  8    0.0    0.0    0.0  0.0  0.0
task stealth  7  9    0.0    0.0    0.0  0.0  0.0
task stealth  7 10   50.0   50.0   50.0  0.0  0.0
task stealth  7 11    0.0    0.0    0.0  0.0  0.0
task stealth  7 12    0.0    0.0    0.0  0.0  0.0
task stealth  7 13    0.0    0.0    0.0  0.0  0.0
task stealth  7 14  100.0    0.0    0.0  0.0  0.0
.     .    . .    .       .       .   . .
task stealth  7 25  100.0    0.0    0.0  0.0  0.0
task attack  7  1  1 1 1 1 1 1
```

Figure 5-13: Example of silkworm task (ground) class command

The Silkworm threat acted much like the artillery threat, appearing in a fixed land location, holding for a set period, then "flying" to the target (either the CVBG or the ARG penetration zones) in order to score a hit. To do so the silkworm threat had to cross the boundary between the land area and ocean area. If an air task had been used there would have been no problem, but we desired that the Silkworm be viewed as a ground threat (i.e. a silkworm launcher), so it needed to be a Ground task type. This Ground type silkworm threat, however, could not travel outside the land boundary, so the spawned Silkworm threat needed to be an Air type task. Because of this land/sea boundary issue two task classes were actually used - one (ground task class) for the initial ground

73

detection and one (air task class) that was spawned, or "launched", upon the
disappearance of the first. The spawning implementation is described in a later section.
A description of the air task class silkworm is shown in Figure 5-14.

```
# task 22: Air contact (silkworm anti-ship missile battery)
task general 22  A  SWA  0.99  0.211  2  silkworm.icon
task mean  22  15.0  20.0  0.00  0.00  0.00  0.00  0.00  5.00  0.00 1.00
task view   22 2 2 2  2 2 2
task stealth 22  0  100.0 100.0 100.0 100.0 0.0
task stealth 22  1    0.0    0.0    0.0    0.0 0.0
task stealth 22  2    0.0    0.0    0.0    0.0 0.0
task stealth 22  3    0.0    0.0    0.0    0.0 0.0
task stealth 22  4    0.0    0.0    0.0    0.0 0.0
task stealth 22  5    0.0    0.0    0.0    0.0 0.0
task stealth 22  6    0.0    0.0    0.0    0.0 0.0
task stealth 22  7    0.0    0.0    0.0    0.0 0.0
task stealth 22  8    0.0    0.0    0.0    0.0 0.0
task stealth 22  9    0.0    0.0    0.0    0.0 0.0
task stealth 22 10   50.0   50.0   50.0    0.0 0.0
task stealth 22 11    0.0    0.0    0.0    0.0 0.0
task stealth 22 12    0.0    0.0    0.0    0.0 0.0
task stealth 22 13    0.0    0.0    0.0    0.0 0.0
task stealth 22 14 100.0   50.0   50.0    0.0 0.0
    .      .  .      .         .      .   .
task stealth 22 25 100.0   50.0   50.0    0.0 0.0
task attack 22  1  1 1 1 1 1 1
```

Figure 5-14: Example of Silkworm task class (air) command

To facilitate the use of a national or CJTF asset to conduct positive identification
of the silkworm launch site a "positive ID attribute" was included in the attribute vector.
This attribute was only readable by the SR7 reconnaissance asset held by the CJTF. To
increase competition among units the "ground" Silkworm could only be engaged with VA
attack aircraft) owned by the CJTF. This was accomplished by again tailoring the

attributes of the task to the resources of the specific platform. Once "launched" the (air) silkworm threat could not be engaged by any platform.

2. Spawning

When initially written the software had no provisions for a sequential game nor one that allowed the simulation to "react" to various decisions of the team. It quickly became apparent that there was a need for a "spawning" command that allowed the designer to present the team with a task that when accomplished would produce another task, such as taking of a hilltop then the holding of the same piece of ground.

The "launch" or spawn of the air task silkworm threat was accomplished using the *task spawn* command. An example of the command format is shown in Figure 5-15.

```
Format:  task spawn  id  num  type  sid(1)  ...  sid(5)
                                     sid(6)  ...  sid(10)
                                     sid(num-4) ...  sid(num)
         # id:    integer id of 'spawner' task for which new tasks will be spawned.
         # num:   integer number of tasks to be spawned.
         # type:  character A or D denoting a task Attack or task Disappear event.
         # sid(I): integer ids of 'spawned' tasks.

NOTE:  i)   0 < num < n, where n is specified via number_of tasks command.
       ii)  a task can be both spawned and a spawner (ie, recursive), however, a task cannot be
            spawned by more than one spawner.
       iii) to prevent spawning cycles, ie, X spawns Y spawns X, we require id < sid(I), I = 1..num.
       iv)  at most 5 spawned task ids specified per line, and, for num > 5, only last line can
            have <= 5 task ids specified.
```

Figure 5-15: Example of Silkworm task class (air) command

For the silkworm transition from ground to air the spawn was keyed to the disappearance of the ground task. If the ground task silkworm was not properly engaged within the wait

75

time then after a set period the air task would spawn, or "launch" and target the CVBG or the ARG. This spawn command is shown in Figure 5-16.

```
task spawn  286  1  D  289

# task 286 is ground task silkworm
# task 289 is air task silkworm
```

Figure 5-16: Example of Silkworm task class (air) command

3.   Stealth command usage

The *platform range* command provided the mechanism for assigning sensor ranges, such as detection and identification. However, this command allowed for setting platform sensor ranges only by class type - air, sea and ground. After designing portions of the scenario it became apparent that the *platform range* command did not provide the flexibility needed to abstract the more complex scenario requirements. Because of the great number of assets and threats being uniquely represented, a method was needed to allow specific task class to specific platform class range assignments. This was implemented using the *task stealth* command. The stealth command resets the detection, measurement, identification, attack and be-attack ranges for a specific task class vis-a-vis a specific platform class. (Overrides default values.) This allows the scenario designer to tailor ranges down to an individual threat or mission, which can be used to create the possibility of a late detection or non-detection of a unique threat. A description of the command is contained in Figure 5-17.

```
Format: task stealth id1 id2 [sensor(1) sensor(2) sensor(3) attack(1) attack(2) ]
        # id1:     integer number, identifying the task class.
        # id2:     integer number, identifying the platform class.
        # sensor(i): floating point numbers:
                i=1: overrides sensor detection range r(j,1) of platforms of class id2 when
                        encountering tasks of class id1 (that are of type j)
                i=2: overrides sensor measurement range r(j,2) of platforms of class id2 when
                        encountering tasks of class id1 (that are of type j)
                i=3: overrides sensor identification range r(j,3) of platforms of class id2 when
                        encountering tasks of class id1 (that are of type j)

Default: values r(j,i) as specified in platform range command -- EXCEPT if threat_flg = 2 or 3, in
which case default values = r(j,5)

        # attack(i): floating point numbers:
                i=1: overrides associated attack range r(j,4) of platforms of class id2 when
                        attacking tasks of class id1 (that are of type j).  Default: value = r(j,4),
                        EXCEPT if threat_flg = 2, in which case default value = 0.0
                i=2: overrides associated be-attacked range r(j,5) of platforms of class id2 when
                        encountering tasks of class id1 (that are of type j). Default: value = r(j,5)
for
        all values of threat flag.


                    Figure 5-17: Example of task stealth command
```

a.   Mines

One application of the *task stealth* command was in the modification of

the mine tasks, both sea and ground. The scenario OPORDER stated that mines were in

the area of battle, but did not specify the location or number. The stealth command was

used to set the detection range of several of the platforms to a fraction of their "normal"

ground ranges. The be-attack ranges were also significantly reduced. The mines would

appear on the display only after a platform crossed into the small detection range. In

some cases the DM had very little time to react to the threat. The speed of the platforms

and the small difference in ranges resulted in a requirement for an immediate response by

77

the DM controlling the affected platforms. An example of the command being used for

the ground mine threat is shown in Figure 5-18.

```
task general 8  G MN  0.01  0.211  2 mines.icon
task mean 8  5.0  10.0  0.00  0.00  0.00  0.00  2.00  0.00  0.00 1.00
task view  8 2  2  2  2  2  2
task stealth  8  0  100.0 100.0 100.0 100.0 02.5
task stealth  8  1   0.0   0.0   0.0   0.0 0.0
task stealth  8  2   0.0   0.0   0.0   0.0 0.0
task stealth  8  3   0.0   0.0   0.0   0.0 0.0
task stealth  8  4   0.0   0.0   0.0   0.0 0.0
task stealth  8  5   0.0   0.0   0.0   0.0 0.0
task stealth  8  6   0.0   0.0   0.0   0.0 0.0
task stealth  8  7   0.0   0.0   0.0   0.0 0.0
task stealth  8  8   0.0   0.0   0.0   0.0 0.0
task stealth  8  9  00.0  00.0  00.0   8.0 02.0
task stealth  8 10   0.0   0.0   0.0   0.0 0.0
  .     .      .  .      .       .      .      .   .
  .     .      .  .      .       .      .      .   .
task stealth  8 23   0.0   0.0   0.0   0.0 0.0
task stealth  8 24   4.0   4.0   4.0   0.0 01.0
task attack 8   1  1 1 1 1 1 1
```

Figure 5-18: Example of task stealth command used in the mine task class

In this example platform 9 and 24 are assigned new ranges for task 8, the

ground mine. These ranges will override the ranges set in *platform range*. The detection

of the mine threat by platform 24, the ground units of MEU2, will occur at four miles,

with the be-attack or "lethal" range of the mine set to one mile. All other platforms used

in the scenario were assigned ranges of 0 to ensure that the mines would not be detected

inadvertently by another asset, such as an aircraft passing over the area.

D. SPEED AND MOVEMENT

The playing area, a 100 by 100 mile square grid, was chosen for a number of reasons, not the least of which was the usability of the display. If the game area were to be expanded much beyond 100 miles the DMs would be forced to "zoom in" much more frequently. It was decided that for the phase one experiment, display confusion (or a lack of user friendliness) should be avoided whenever possible. For this experimental reason the size of the display was chosen simply based on what looked the best - and allowed each DM to see the entire battle space without excessive command requirements.

To place a land mass, ocean area, full carrier battle group and an amphibious readiness group in such a small area did pose some interesting problems. Realistically, no carrier battle group is going to be within 20 miles of a hostile coast, at least not until air superiority is achieved. This simulation required some constructive time/distance "changes" to make the assets behave more like the real world. In other words, the movable elements of the scenario, both assets and threats, moved the same as in the real world - in relation to each other .

It was necessary on occasion to adjust the speed of the targets and assets to allow the various missions to be accomplished within the 40 minute simulation window - in some cases the scenario speeds of some units was triple the "real world" speed. There did not appear to be any real problems with this modification in terms of scenario believability. An example of the need to change speeds was the observance (in trial runs) that in one of the scenarios the southern MEU was always arriving at the objective first,

which avoided the competition that we were trying to induce. By slowing the southern MEU's speed down by just 5% we were able to ensure that both MEU's reached their objectives at roughly the same time.

E. EXPERIMENTAL CONTROL

After the first round of training runs with the actual teams the length of the scenarios was fixed at 40 minutes. Several times during the training runs the teams would "finish" i.e. reach the objective/goal, prior to the end of the game clock. For the purposes of data taking and to support the controlled conditions of the laboratory environment the teams were required to remain at the consoles and perform any game elements until the clock reached 40 minutes. The simulation lacked a sense of realism when this situation occurred. A mechanism was needed to terminate the game following the completion of the mission objectives. This mechanism was implemented using the *end game* command. The end game command, shown in Figure 5-19, allowed the designed to specify the final mission task as a trigger for the termination of the game.

Format:	game end id time # id: game will gracefully end after the task number id has been attacked. # time: float delay time to wait before ending the game. Default: id=399, time=5.0.
Example:	game end 390 5.00

Figure 5-19: Example of end game command

VI. DDD-III EXPERIMENT PHASE I - FUTURE ISSUES

Many challenges were encountered in modeling the Joint environment utilizing the DDD-III during the planning and development of the experiment. Many of these challenges were solved prior to the experiment, in an evolutionary manner, whereby various abstractions were developed, tested and installed in the scenario. No "show stoppers" were encountered, although several minor elements of the scenario, such as the number of NSFS ships available, were modified to be more easily represented by the DDD. These small modifications of the scenario, however, were transparent to the decisionmaking teams. As such the basic premise of the experiment and the general hypothesis were unaffected by these changes.

While implementing the current scenario, additional ideas for abstracting elements within the DDD-III were considered for future implementation. Some of these concepts have been developed - others are still in the "idea" stage. The major concepts for future development are discussed below. (A discussion of lessons learned from the phase one experiment is also contained in additional documents [Ref 8].)

A. INFORMATION STRUCTURE

1. Information net design

The information structure currently available in the DDD-III is of a single "net" design. Data is available to all decisionmakers as long as the individual platform sensor

is capable of reading the data. The scenario designer has a limited ability to tailor the information received by the individual DMs. Some degradation of the data is possible via the various "noise" inducing commands, but again, this data must be readable by the platform sensor. The Joint information structure in the real-world, however, is composed of a multitude of information nets, separated by function, geography and mission need.

A true abstraction of the Joint environment would give the designer the ability to provide different nets to different DMs. A representation of a general information net structure concept is shown in Figure 6-1.

To implement this concept and accomplish the functions required for a net structure the *task view* command will be replaced with three new commands:

- *Number_of nets nets*

- *platform nets*

- *access nets*

Figure 6-1: sample net structure
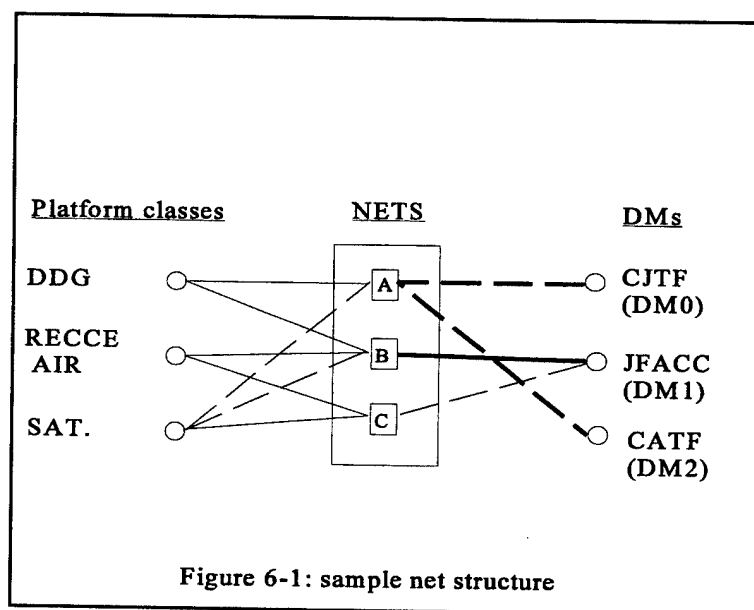
The first command, *number_of nets nets* will specify the number of information nets that exist in the scenario. The designer will then customize the connections between the platforms and the nets in order to determine the type of information available from sensors on a particular platform class. This connection can be representative of a low bandwidth device, in which case the sensor may report only detection, or a highband device, in which case the sensor may provide measurements of attributes.

Finally, the designer will specify the connection between the nets and the DMs, determining what information will be presented to the DMS. The connection will specify the type of information desired: Air, Sea, Ground, any combination of the three.

This method of implementation will allow for greatly increased flexibility in the information structure and more closely model the real-world. This net structure concept can be built upon to further enhance the information distribution throughout the scenario. However, there are many procedural concepts, such as: (1) the use of filters in the connecting lines, (2) who controls the assignment of nets and (3) who can join a net, that require further exploration.

2. Information growth

In the current version of the DDD, information presented to the DMs, in the form of sensor measurements from platforms, changes only with distance. (i.e. the accuracy of the sensor can be set to be range dependent) This creates a more accurate representation of sensors available in the real-world. However, there is more that could be included

with respect to the manipulation of data change. Information in the real-world does not remain stagnant but changes with time as the conditions of the environment and of battle change.

One element is that of a dynamic information environment. As the scenario progresses, information on a particular task can change to represent an increased threat or a capability that may not have been initially detected.

3. Communications

a. Internal message traffic

The current version of DDD-III allows for message traffic among the DMs, although messages are limited to menu selected and pre-scripted options. During the course of the experiment, it became apparent that the menu selected messages were not sufficient for a high-level decisionmaking simulation. The limited number of messages constrained the flow of command and control information among the DMs, resulting in player frustration.

This change is in the design phase. Whatever the method of implementation, there will be significant requirements on the recording of message traffic for the purposes of data analysis. The improved message system must at a minimum provide to the DDD log files the text of the message, the time sent, sender, receiver and any DMs that where included in the message distribution.

b.   External message traffic

An external message is informational text, originating outside the DDD

software, which must be delivered to any or all DMs. An outside message is

normally an intelligence report but can be anything since no restrictions are

placed on its content. During the scenario runs the outside messages were in the form of

paper copies handed to the DMs by experiment monitors. This was a rather cumbersome

process, requiring coordination with the scenario time line, as well as the administration

of all the various messages to be presented to four teams on four scenario runs.

Future DDD-III scenarios will present "outside" messages via the message display

system contained in the software. The scenario designer will specify message delivery

based on time or event, and will be able to specify the desired distribution.

B. COMMAND STRUCTURE

1.   Level of decision making

The tasks presented to the DMs were all handled in the same manner for this

experiment. Each DM followed similar procedures to accomplish a presented task. In

the real-world, however, mission objectives vary based on the relative level of the DM

involved, as well as how the objective is actually processed. For example: the objective

of taking the beach represents a single mission to the CJTF. For the component

commander (GCC), however, this objective involves many smaller objectives, such as

picking the beachhead, placing troops into position, assigning covering air, etc.. The

same broad task may appear to be very different to different levels of the same decision

team. This difference requires that separate parts of the team perform tasks that may be viewed as fundamentally different.

One method of integrating this difference into the simulation may be to create a set of tasks that present a series of "subtasks" to be accomplished by DMs of varying decision levels. Unlike the *prerequisite* command, this set of "subtasks" would reside on a single task, be presented together, and not necessarily require that the "subtasks" be accomplished in any given order.

C. ENVIRONMENT

1. Geography

The DDD-III currently uses "tasks" to represent various geographic obstacles, such as swamps, hills and towns. Additional environmental commands added to the general commands section would simplify scenario design, such as preformated swamps, hills, roads, etc. These environmental commands would be used to increase the richness of the visual display. To represent an objective to be taken by the DMs the definition and representation of a task class would still be required.

D. PLATFORMS

1. Ownership

During the scenario, ownership of subplatforms developed into a significant issue. The partial resolution was development of the *platform subplatform* command structure,

which, among other elements, specified the ownership of a subplatform following its "launch". (see chapter V)

There are situations in the real-world in which two identical assets, positioned on the same parent platform, are owned by different Decisionmakers. An example of this for our scenario was NSFS - both MEU's needed to simultaneously "own" fire missions of the same subplatform class (5in) on the same ship (DDG). With this situation the challenge for the designer was to work around the DDD-III, which would not allow identical subplatforms to reside on the same platform with different owners. This limitation was averted by the addition of another DDG to carry the second MEUs NSFS missions.

The *platform subplatform* command will be modified to remove this limitation. This will reduce the number of platforms required in the scenario and more accurately represent the real-world environment.

2.   Attrition

This scenario was designed so that assets needed for mission accomplishment could not be destroyed, either by team error or by random chance. Attrition was represented by the loss of team strength - a score maintained for a final measure of success of the team performance. It is generally undesirable, from an experimental point of view, for available team assets to be a function of the team's path through the scenario. Some attrition, however, would definitely be a positive step towards real-world combat

situations. The method of implementation and the resulting effect on the experimental conditions must be further researched.

3. Loss of resources (damages) to an asset

Closely related to attrition is the concept of mission capable platforms. One mechanism to abate the attrition problem would be to allow damage to be sustained by platforms. The amount of damage could be limited to a final mission-representative figure, or "bottom line". The implementation of this concept could result in the need for more assets for a given task to be accomplished.

4. Auto-stop (intelligence)

The current version of DDD does not contain any mechanism for a platform to "react" to a threat. The *stealth* command, used frequently in this scenario, resulted in several tasks "popping" up within a very short range of the platform. The platform, unaware of the task in its path, would of course continue on the course of action last ordered by the DM in control. In the real-world, however, upon detection of a threat or anomaly, an asset would stop and reassess the situation to determine whether new action is needed, before proceeding.

To implement this "intelligent" response to a "pop-up" threat, the software will be modified to stop a platform upon detection of a task in its path, and possibly to engage the task. The specific issues of the option have yet to be determined, but there will be a tailored set of task parameters to trigger this "autostop" concept.

E. TASKS

1.   Arrival based on location vs time

Currently, all tasks are time dependent - the scenario designer scripts the sequence of tasks and then uses a time line to designate arrival times. The task spawn command is one attempt to replace the time dependency with situational dependency, but this is only the first step. In the scenario, designers were concerned about maintaining the advancement of forces at relatively the same pace, ensuring that hoped for competition events would occur. The ability to have tasks sensitive to "trip wires" would have been a great advantage in controlling the advance. For example - tanks that appeared on the road were carefully timed and placed so as to "appear" to both ground forces simultaneously. However, in some scenario runs the beach landings did not go according to the schedule, and tanks appeared on the road long before the forces were actually in a position to be engaged by them.

2.   Dynamic tasks (attributes, etc., as a function of time)

It would have been interesting if the attributes of the enemy ground units changed as the scenario progressed. Thus, depending on when the task is actually engaged by the DMs assets, the complexity, resources, information and coordination elements may be significantly altered. For example, in this scenario, enemy tanks required only one section of aircraft to engage and destroy them. Future design considerations include consideration of changing task attributes. What if, however, the attributes of the tanks changed with time, i.e. in a period of time the tanks are "reenforced" with additional

forces? They then would require addition assets to destroy them. For example two sections of combat aircraft, further complicating the resource allocation issues faced by the DMs.

3.   Parallel attacks by several DM's (done)

Currently, a form of parallel attacking is possible - during the scenario there were several tasks that required more than one platform's assets to effectively engage and destroy them. This method of attacking was very constrained in the sense that the amount of resources required to attack was known and the platforms engaged simultaneously. Balancing the desired realism with the practical elements necessary to force asset competition virtually forced a unique task to asset association [Ref 6]. This association is, of course, present in the real-world but in a much more flexible manner. For instance, a more generic requirement to engage aircraft with an anti-air weapon, vice having the specific weapon designated ahead of time.

One method proposed to circumvent the unrealistic asset-task assignment is to allow for simultaneous attacks by platforms. The DMs would be aware of the minimum required resources but have the flexibility to achieve these resources by combining the platforms in any desired manner.

4.   Unknown resources needed for engagement

In the current DDD-III the resources required to properly engage a task are reported to the DMs provided there is a platform sensor within range of the task able to

measure the attributes. The ability to consistently view the match-up prior to an engagement is something that does not exist in the real-world.

A more realistic representation would be for the attacking platform to have little or no idea as to the resources required. For instance the DM is required to engage the threat but does not know if he will be successful. Once the platform engages the task the attribute to resources match-up would be displayed. The DM might then have to call upon other platforms, either organic or non-organic, to provide the necessary resources to make up the difference and complete the attack. This concept would be effectively supported by a new parallel attack command.

5. Tasks reduced in ability following an attack

In the current DDD-III all tasks, if engaged properly with platforms holding the required minimum resources, are destroyed. The capability to reduce or degrade task effectiveness verses actually destroying the task (i.e. a mission kill) would allow the simulation to more accurately model the real-world. Many times ROE requires that a reasonable and appropriate response be used by the subordinate commanders in the execution of their mission.

6. Tasks warned off vice destroyed

No mechanism currently exists to warn off a task vice engaging and destroying it. The realities of the real-world are seldom this final - there are rules of engagement to follow, plus international law and convention to abide by. In many combat areas, aircraft

and surface vessels are routinely warned off by the forces in the region, presenting the opportunity for the unknown contact to leave the area before being engaged.

F. SPEED AND MOVEMENT

1. Variable task speed

Much of the movement of both task and platforms is linear. Simple grid locations are presented by the designer to the scenario generator, which translates these XY coordinates into motion vectors for the tasks. No mechanism exists to allow variation or to simulate a responsive target - the task speed is currently set to a time line. If the ability existed to speed the task up when a platform came into close proximity, the threat would be more representative of a dynamic real-world threat.

VII. SUMMARY

A. PROJECT CONCEPT

A revolution in Command and control (C2) capability, brought about by new communications technology and advanced computer capabilities, has created the "Global Awareness" concept. How Decisionmakers (DMs) adapt their organization to coordinate information, resources and activities to fulfill their mission, within the realm of Global Awareness, is the subject of current research efforts.

The Office of Naval Research has commissioned a research effort into this far reaching topic - the Adaptive Architectures for Command and Control (A2C2) project. The project required a mechanism to abstract "real world" problems into a controlled laboratory environment where a variety of experimental conditions could be manipulated.

A new program was needed to support this model-based empirical research. The result was the development of the Distributed Dynamic Decisionmaking (DDD-III) paradigm.

B. DISTRIBUTIVE DYNAMIC DECISIONMAKING (DDD-III)

1. Requirements

A user friendly, easily modified, "low level" command and control simulator, the DDD-III paradigm models the Joint real-world environment. A wide range of commands and data strings are available to abstract the complex components and variables

associated with a Joint battle space. The DDD-III was developed to fulfilling the following needs:

- Create a "realistic" Joint environment within a computer simulation, abstracting the various elements of Army, Navy, Air Force and Marine warfare, found in the Joint Task Force environment, to an understandable computer representation.

- Provide for the easy manipulation of key structural variables to allow testing of basic hypotheses dealing with structural change.

- Avoid the need to build/require a technical domain expertise in test subjects, thereby allowing the experiment to be conducted on-site at NPS using available officers.

- Create a simulator that would allow for ease of scenario design/change, data collection and data retrieval DDD-III provides a multi-player, real-time, model-based simulation environment.

2.   Environment

The DDD-III has the ability to explicitly operationalize many of the relevant organizational dimensions within a Joint task force (JTF). There are three essential factors which make the DDD-III ideal for the tier-I experiments:

- Joint warfare framework

- Mission structure - represented as tasks, linked together by time, space and/or precedence to provide a theater level <u>mission</u> for the organization to accomplish.

- Force competition - used to "force" coordination between the DMS in setting priorities, allocating limited organic and non-organic resources as to achieve the *overall* mission objective.

3. Core elements

The DDD-III software gives the user the ability to modify dimensions of task, platform and organizational structure in order to test a specific hypothesis. These dimensions are the core elements of the paradigm. The dimensions are manipulated to study their interactions and to elicit possibly changes in organizational structure and response from a team.

C. PHASE ONE EXPERIMENT

1. Hypothesis

The phase one hypothesis can be summarized effectively as follows: "An organization with a common functional commander is better for certain tasks than an organization without one."

2. Goal

The phase one experimental goals were twofold:

- To test the interaction between task structure, organization structure, and components in a joint environment given a common operational picture across all levels of hierarchy.

- To validate the DDD-III paradigm and the procedures of abstraction used by the scenario designer to develop the scenario specifics.

3. Scenario

The scenario chosen was representation of a "typical" future requirement for US forces, a Joint forces mission to capture a port and airfield which will allow for the introduction of follow-on forces. The assets available to the DMs represented force structure already in use in the real-world Joint environment.

4. DDD Development

One significant challenge was to model elements of the real-world into the DDD-III without extensive software modifications or complex commands, consuming the scenario designers time and distracting from the goal of the simulation.

a. Phase one issues

Many unique applications of the generic DDD-III commands were developed in order to represent the Joint world as accurately as possible. The challenge

was to abstract elements of the real-world that did not fit the generic mold of a direct threat, such as geography and stand-off weapons. Major implementation and abstraction were discussed.

b. Future issues

While implementing the current scenario requirements, additional ideas for abstracting elements in the DDD-III were considered for future use. Some of these concepts have been developed while others are in the "idea" stage. The major concepts were reviewed and the possible changes to the DDD discussed.

D. FUTURE EXPERIMENTS - PHASE TWO

Although analysis of this experiment is not yet completed, initial indications are that the DDD-III provides the needed abilities for the modeling of the Joint warfare environment into a tier-I level experiment. The results of this experiment, coupled with modeling theory being developed separately, will form the basis for phase two. It is anticipated that these experiments will involve changing of organization structure, in response to changing task structure, during the simulation run. This phase is expected to be performed in late 1996.

E. CONCLUSIONS

The underlying DDD-III paradigm was found to be valid for tier-I experimentation - the DDD-III allows the scenario designer to easily manipulate and control the many

experimental dimensions. Additionally, the experiment fulfilled its two general purposes:

- To test the interaction between task structure, organization structure, and components in a joint environment given a common operational picture across all levels of hierarchy.

- To validate the DDD-III paradigm and the procedures of abstraction used by the scenario designer to develop the scenario specifics.

Finally, this experiment provided much needed data and lessons learned to support the next experimental phase.

APPENDIX A. [DDD Scenario Generator Users Manual]

1. INTRODUCTION

2. GENERAL ITEMS

3. CLASS INFORMATION

      3.1 PLATFORM CLASS INFORMATION

      3.2 TASK CLASS INFORMATION

4. STATE INFORMATION

5. MANEUVER INFORMATION

6. SETTING UP AN EXPERIMENTAL SPECIFICATION FILE

7. EXAMPLE FILES

1. INTRODUCTION

The 3rd-generation Distributed Dynamic Decisionmaking (DDD-III) paradigm was designed to meet the needs for empirical research in adaptive architectures for Joint Command and Control (C2). The DDD-III is implemented as a multi-player, real-time simulation that provides a team of decisionmakers with an air, sea and ground environment, a variety of task classes, and controllable platforms with subplatforms, sensors and weapons (resources). This flexible research paradigm provides the ability to conduct controlled experiments in a laboratory environment, using problems that are abstractions of the "real world".

The design of the DDD-III focuses on the dynamic/execution phase of the mission and allows for manipulation of key structural variables in task and organizational dimensions. The DDD-III has the ability to constrain and/or to manipulate organizational structures such as authority, information, communication, resource ownership, task assignment, etc.

The scenario generator is the tool through which the scenario designer translates the mission requirements found in the "real-world" Joint military environment into DDD-specific constraints, defining the Joint warfare "game" world. The scenario generator assists the experimenter in preparing the many options and variables used to create the game. There are four major categories of variables available to the game planner to create the desired simulation:

1) General information - including time, number of players, shape of land, etc.
2) Platform and Task class information - defining both friendly and "hostile" forces.
3) State Information - defining attributes, resources and game element sequencing.
4) Maneuvering information - defining the actual movements of the tasks.

2. GENERAL ITEMS

The commands in this section define the structure and overall generic features of the scenario. These commands allow the scenario designer to create a "framework" within which the Joint environment can be abstracted. The chain of command, number of decisionmakers, game duration and other structural elements are enumerated, as well as land, roads, boundaries and other geographic features. Additional features can be added by tailoring task commands (found in section three) to represent fixed areas to be avoided by the simulation players. The default values of each parameter when applicable are listed within the command description.

Decision Structure
This command defines the number of decisionmakers present in the simulation. In

addition it identifies the command structure they are contained within. (in reference to the other decisionmakers) This allows the scenario designer to specify command structure, which affects authority/responsibility and asset control issues among decisionmakers. Only acyclic structures are supported in DDD-III.

Format: decision structure ndm
           name(0) name(1) ... name(ndm-1)
           cd(0) cd(1) ... cd(ndm-1)
    # ndm   : number of decisionmakers in the game. default=4
    # name(I) : alphanumeric string (up to 5 chars) for DMi's name. default='DMi'
    # cd(I)   : the dm to whom DM#I reports (DM#I 's boss). default=0

constraint : cd(I) ≤ I, to assure that a decisionmaker will only have one "boss" (Acyclic
              structure)

Example: decision structure 6
           CJTF   GCC   MEU1   MEU2   CVBG   ARG
           0   0   1   1   0   0

Explanation: There are six decisionmakers defined. DM1 is assigned the name CJTF, DM2 is assigned the name GCC, etc. The command structure in this example is a three tiered hierarchy consisting of the CJTF as overall commander, with GCC, CVBG and ARG reporting to the CJTF. The two MEU's are reporting to GCC.

Simulation Time
This command sets the scenario duration in seconds, thereby allowing the experiment designer to determine the maximum game time for each scenario run. The *end game* command can be used to stop the simulation if the decisionmaking team has fulfilled their objectives prior to reaching this preset simulation time.

Format: simulation time ts
    # ts : floating number. Maximum time allowed in the game. default=600.0

Example: simulation time 2400.0

Explanation: The game will last a total of 2400 seconds (40 minutes)

Number of Tasks
This command sets the total number of tasks allowed during the simulation. A task is used by the scenario designer to represent enemy threats, mission objectives, fixed land objectives, etc. Each task must be further defined in the state and maneuver sections - if

not the scenario generator will automatically, <u>randomly</u>, create tasks to reach this preset number. (Which can significantly impact the scenario sequence)

Format: number_of tasks n
      # n : integer number of tasks in the game, max=200, default=50

Example: number_of tasks 140

Explanation: A maximum of 140 individual tasks exist in this simulation. (Not all tasks may necessarily "arrive". This depends on their arrival times and the simulation time)

<u>Simulation Scale</u>
This command defines the "size" of the playing area. The value specified will represent the length of each side of the display area in miles. (The units in the simulation can be miles, Km, ft, etc, - as long as consistency is maintained throughout the scenario file specification) The display area will be overlaid by a ten by ten square grid to provide map coordinates.

Format: simulation scale s
      # s : floating number. default=1.0

Example: simulation scale 100.0

Explanation: The playing area will be a 100 by 100 square mile grid area. (The XY grid used in the display is as follows: X coordinate begins from 0 on the left side of the screen and goes to 100 on the right. The Y coordinate begins from 0 on the top of the screen and goes to 100 on the bottom.)

<u>Number of Attributes</u>
This command sets the total number of attributes used to define the characteristics of the tasks. Each task attribute in the vector is assigned a unique name. The attribute value position in the vector is critical - later inputs or references to the attribute vector rely on the order of the numerical values to set the attribute characteristics. (ie: all attributes must be defined each time when referencing the attribute vector) The first two attributes in the vector are fixed in definition: attribute one = Value, is the value that the task is worth to the decisionmaker when properly accomplished. (It also represents the amount lost if not successfully attacked.) Attribute two = Time, is the duration assigned to the engagement when the task is prosecuted (actually attacked) by a platform.

Format: number_of attributes natt
              name(1) name(2) ... name(natt)

natt   : integer number up to 20. Default=3
name(I) : string of up to 5 chars for attribute#I's name. Default='Attri'

Example: number_of attributes 10
               Value Time Air Sea Groun hold Mine tank Med enemy

Explanation: A total of ten attributes are defined for the attribute vector. Each attribute is assigned a name: Value, Time, Air, Sea, etc, whose position in the vector is unique..

Number of Resources
This command sets the total number of resource categories that will be used to define the characteristics of the platforms. A platform is used by the scenario designer to represent friendly assets, such as aircraft, ships, ground units and fixed bases. Each platform will have a vector of resources where each element in the vector is assigned a unique name. The position of the resource value in the vector is critical - later inputs or references to the platform resource vector uses the order of the numerical values to set the resource characteristics. (i.e.: all resources must be defined when referencing the resource vector)

Format: number_of resources nres
               name(1) name(2) ... name(nres)
   # nres   : integer number up to 20. default=3
   # name(I) : string of up to 5 chars for resource#I's name. Default='Resi'

Example: number_of resources 7
               air sea ground hold mine tank Med

Explanation: a total of seven resource categories are defined in the resources vector. Each resource is assigned a name; air, sea, ground, etc., whose position in the vector is unique.

Renewal Interval
This command allows the user to preset a screen renewal interval in order to avoid processor overload, caused by the computers' attempt to continuously update all platform and task positions to all screen displays.

Format: renew interval t
   # t : floating number of seconds between clock and screen updates. Default=1.0

Example: renew interval 0.5

Explanation: The screens will be updated every half second, which will be sufficient in a basic simulation of six players - however, a more involved game that has a greater number

of players may require a longer refresh cycle, such as 2 seconds, depending on the CPU.

Random Seed

This command sets the random seed used in the simulation run. The random seed is applied to produce a sequence of pseudo random numbers that are used to generate task and platform information. For the same xsnnnn.dat file, different random seeds will produce different information elements on the tasks and platforms. This function can be used by the scenario designer to provide a randomness for the values of the attributes. (This randomness can be useful in defining the attributes of the neutral tasks to provide some diversity between scenario runs)

Format: random seed r
    # r : positive integer number. Default=5

Example: random seed 1

Explanation: Provides a random seed value of 1 to the random noise generator

Message Number

This command sets the number of information channels each decisionmaker owns. A channel can hold only one message at a time. If all channels owned by a decisionmaker are filled with messages the decisionmaker cannot send out any messages until a channel is cleared.

Format: message number n
    # n : positive integer number of communication channels available. Default=4

Example: message number 4

Example: Assigns a total of four message channels to each decisionmaker.

Communications Epoch

This command sets the time a communications channel is held for transferring a message. Unlike *communications delay*, Which sets the time delay for a message to reach its destination after being sent from the decisionmaker, *communications epoch* concerns the time the communications equipment itself is tied up during message transmission. This function can be used by the scenario designer to simulate the internal delays associated with routing a message to be released through your own command hierarchy.

Format: comm epoch t
    # t : floating number of seconds a comm. channel is busy after sending a msg,

Default=6.0

Example: comm epoch 6.0

Explanation: Any communications channel used by the decisionmaker will be tied up for a period of six seconds following transmission.

Communications Delay
This command sets the time it takes for a communication message to reach its destination after initiation from the source decisionmaker. This command allows the scenario designer to account for the delays associated with message transmission in a global perspective (satellite delays, system busy signals, etc)

Format: comm delay t
    # t : floating number of seconds it takes for a message to be received. Default=15.0

Example: comm delay 5.0

Explanation: Messages will be "delayed" for a period of five seconds before being displayed on the receiving decisionmakers terminal.

Request Delay
This command sets the time it takes for a resource request message to reach its destination. Functions similarly to the *communications delay*. This command can be used by the scenario designer to simulate the delay encountered when a request for assistance is sent to another command. A zero delay would mean instantaneous response.

Format: request delay t
    # t : floating number of seconds it takes for a request to be received. Default=15.0

Example: request delay 5.0

Explanation: All resource request messages will be delayed a total of five seconds before being displayed on the destination terminal.

Transfer Delay
This command sets the time it takes for a transferred platform to reach its new owner after the initiator has completed the transfer command. This command can be used to simulate the time required for the chain of command to assign that unit to the new task force.

105

Format: transfer delay t
    # t : floating number of seconds it takes for a transfer to be received. Default=20.0

Example: transfer delay 10.0

Explanation: Platform transfers will be "delayed" a total of ten seconds before being available to the receiving decisionmaker.

Attack Delay

This command sets the *default* time it takes for a task to be destroyed once an attack has been initiated. Both the attacking platform (s) and the task are frozen for this period. This allows the scenario designer to simulate the use of assets in a real-world engagement, where engagements "remove" the asset from the order of battle for a period of time.

Format: attack delay t
    # t : floating number of seconds an attack requires.

NOTE: Since the 2nd attribute is the attack delay, this command sets the default for the
        2nd attribute. Default=15.0

Example: attack delay 15.0

Explanation: The default attack delay (freeze time) for any attack is set to fifteen seconds.

Scramble Identification Number

This command sets the task identification number scramble flag. Task numbers are specified in the maneuver section and are displayed as part of the task identification to the *display* screens. When set, this flag renumbers the display identifications. This command allows for the identification numbers of the contacts to change with each run of that specific scenario, which prevents the team from anticipating tasks based on their identification number.

Format: scramble id d
    # d : integer number. Default=0
    #   :0 -- do not scramble the task id
    #   :1 -- scramble the task id

Example: scramble id 0

Explanation: Task numbers are not scrambled and are as specified in the maneuver section.

Communications Permission
This command sets the communications structure for the decisionmakers within the simulation. If a communications link exists from decisionmaker i to j, DMi can request/send resources, information and action/intention messages to Dmj. This command is one way in which the scenario designer can augment the chain of command in the simulation, forcing DM's to follow preset communications routes.

Format: communication permission
           P(0,0)   P(0,1)   ... P(0,ndm-1)
           P(1,0)   P(1,1)   ... P(1,ndm-1)
           P(ndm-1,0) P(ndm-1,1) ... P(ndm-1, ndm-1)
    # P(i,j) = 1, DMi can communicate with DMj.   (i.e. link exists from I to j)
    #   = 0, DMi can't communicate with DMj.
    # Default: all diagonal elements are 0, the rest are 1

Example: communication permission
          0   1   1   1   1   1
          1   0   1   1   1   1
          0   1   0   1   1   1
          0   1   1   0   1   1
          1   1   1   1   0   1
          1   1   1   1   1   0

Explanation: The matrix defines the communications links between DM's. All decisionmakers can communicate with each other except for DM2 and DM3, who cannot communicate with DM0, and are therefore forced to pass all requests for assets and messages to DM0 through an alternate routing. In our specific example DM2 and DM3 represented MEU's, who were expected to communicate through the GCC to the CJTF. (See *decision structure*)

Land Area
This command sets the portion of the game area defined as land. Sea tasks/platforms cannot cross on to the land area, and ground tasks/platforms cannot cross into the sea area.

Format: land area x1 y1 x2 y2
    # x1 : Upper-left X-coordinate
    # y1 : Upper-left Y-coordinate
    # x2 : Lower-right X-coordinate

    # y2 : Lower-right Y-coordinate
    # Default=no land.
Constraints: $0 \leq x_i \leq s$; $0 \leq y_i \leq s$ (s = simulation scale)

Example: land area 0.00 40.00 30.00 100.00

Explanation: The defined land area will be a rectangle 30 units wide and 60 units long, beginning in the lower left corner and extending upward into the game area display.

Draw Line

This command allows the user to draw lines of varying thickness and length. These lines can be used to mark areas, define roads and add geographic relief to the game area display. Multiple *draw line* commands can be issued.

Format: draw line x1 y1 x2 y2 [width]
    # x1   : Upper-left X-coordinate
    # y1   : Upper-left Y-coordinate
    # x2   : Lower-right X-coordinate
    # y2   : Lower-right Y-coordinate
    # width : (optional) Line width, in pixels. Default: 1
    # Default=no lines drawn.

Example: draw line 5.00 95.00 5.00 75.00 2

Explanation: This will draw a line, two pixels wide, from xy grid (5,95) (upper left corner) to xy grid location (5,75). This line will appear as a straight vertical line 20 units long.

Draw Rectangle

This command allows the user to draw rectangles of arbitrary sizes in the game display area. This command can be used much like the *draw line* command to demark geographic areas, such as towns, etc., as needed in the playing area.

Format: draw rectangle x y w h [width]
    # x   : Upper-left X-coordinate
    # y   : Upper-left Y-coordinate
    # w   : Width
    # h   : Height
    # width : (optional) Line width, in pixels.   Default: 1
    # Default=no rectangles drawn.

Example: draw rectangle 27.00 49.50 2.50 4.00

Explanation: This will draw a rectangle 2.5 units wide and 4 units long, using a 1 pixel wide line, with the upper left corner of the rectangle located at xy grid position (27,49.5).

Draw Circle
This command allows the user to draw arbitrary circles in the game area display. This command can be used much like the *draw line* and *draw rectangle* command to demark geographic areas, such as landing zones, as needed in the playing area.

Format: draw circle x y radius
    # x   : Center X-coordinate
    # y   : Center Y-coordinate
    # radius : Radius of circle
    # width : Line width, in pixels. Default: 1
    # Default=no circles drawn.

Example: draw circle 10.0 50.0 5.0

Explanation: This will draw a circle with a radius of 5 units, using a 1 pixel wide line, centered at xy grid position (10,50).

Penetration Number
This command specifies the total number of penetration zones contained in the game area display, and must come before the penetration zones are defined. (See *penetration zone* command for an explanation of the penetration zone use)

Format: penetration number npen
    # npen: Number of penetration zones.   Default is npen=1, a circular penetration zone
            that will be sensitive to all task classes. If the user fails to give the location of
            the zones they will be clustered at the center of the display area at (s/2, s/2).

Example: penetration number 7

Explanation: Defines a total of seven penetration zones in the game area display.

Penetration Zone
This command defines the actual penetration zones, either rectangular or circular. The zones are numbered starting from 0. The zones will appear in the display as a thick red bordered area.   Penetration zones represent areas that are sensitive to various task classes.

When a zone is penetrated by these tasks the team losses points. This allows the scenario designer to create an area, vice a specific point, of vulnerability that must be protected - i.e.: the area around a carrier in which the enemy has the capability (and desire) to strike with cruise missiles.

Format one: penetration zone rectangle ip x y w h
    # ip: number of the penetration zone
    # x: floating number, the x coordinate of upper left corner of the penetration zone.
    # y: floating number, the y coordinate of upper left corner of the penetration zone.
    # w: floating number, the width of the penetration zone.
    # h: floating number, the height of the penetration zone.

Format two: penetration zone circle ip x y r
    # ip: number of the penetration zone
    # x: floating number, the x coordinate of center of the penetration zone.
    # y: floating number, the y coordinate of center of the penetration zone.
    # r: floating number, the radius of the penetration zone.
    # Default: penetration zone circle, in the center of the screen with radius
            0.2*(simulation scale)

Example: penetration zone rectangle 0 26.50 70.00 4.00 6.00
               penetration zone circle   1 25.00 45.00 5.00

Explanation: Assuming npen=2, the first command will create a rectangular penetration zone, with id=0, 4 units wide and 6 units long with the upper left corner located at xy grid position (26.5,70). The second command will create a circular penetration zone, with id=1, of 5 units radius with the center located at xy grid location (25,45). (The individual task sensitivity to these zones is defined in the *task penetration* command)

3. CLASS INFORMATION

The platform/task class information specified in this section sets the common features of the platforms and tasks, which are in turn used to generate specific platform/task state information.

3.1 platform class information

Platform Classes
This command sets the number of unique classes of platforms that exist in the scenario. For the purpose of platform class number both platforms (parent) and subplatforms (child) are counted equally. (The actual specification of platform/subplatform relationship is set

110

using the *platform subplatform* command)

Format: platform classes npc
    # npc : integer number up to 35. Default=6

Example: platform classes 26

Explanation: A total of 26 classes are available for use in the scenario (for both platforms and subplatforms combined)

<u>Platform General</u>
This command sets the parameters for each *specific* platform class. These initial specifications define the platform and form the basis for platform uniqueness. The values of these parameters and their various combinations provide a broad range of options to the scenario designer to create models of real-world assets. (Air, sea and ground)

Format: platform general id type class mv xfr
                              reu ret e   r d
    # id: integer, uniquely identifying this platform class, starting from 0.
    # type: single character A, S, G for Air, Sea, or Ground vehicles. default='A'
    # class: string of up to 3 characters which gives a "name" to this class of platforms.
          default='Pid' (where id = id# above)
    # mv: maximum velocity of platforms of this class. Default=0.0070*simu_scale
    # xfr: transferability flag:
        = 0 platforms of this class <u>cannot</u> be transferred.
        = 1 platforms of this class <u>can</u> be transferred.
          Default=1
The following variables refer to the platform class when used as a subplatform. They are NOT optional parameters.
    # ret: return ability flag:
        = 0 subplatforms of this class cannot be returned after being launched
        = 1 subplatforms of this class can be returned after being launched. Default=1.
    # reu: reusable flag:
        = 0 subplatforms of this class cannot be reused once they are used in attack.
        = 1 subplatforms of this class can be reused for multiple attacks
          Default=1.
    # e: endurance, floating number of seconds that the subplatform is available for use, once launched. Default=60.0
    # r: refuel time, floating number of seconds. The time between the return of an asset and its availability for relaunch. Default=5.0
    # d: launch delay, floating number of seconds. Default=20.0.

<center>111</center>

Example: platform general 0 A VF 0.9900 1
                              1 1 3000.00 3.00 15.00

Explanation: This defines a platform class with an id=0, as an Air type. Any platform of this class will be named VF, and have a maximum velocity of .99 (1.0 equals one mile per second). Any such platform is transferable, returnable and reusable, with a life duration of 3000 seconds (50 minutes), a refueling time of 3 seconds and a launch delay of 15 seconds. Transferability indicates that the platform can be given to another DM. Returnability indicates that the platform can go back to its original parent. Reusability indicates that the platform will be available for use again after an attack.

Platform Resource

This command assigns the default values of the resource vector associated with that class of platform. These values will be carried through to each platform of that class. This allows the scenario designer to "arm" the platforms with various "weapons", giving the platform the ability to engage tasks.

    Format: platform resource id   w(1) w(2) ... w(nres)
    # id   : integer number, identification of the platform class.
    # w(I): integer number, the amount of resource of category I. Default w(I)=1

Example: platform resource 0   50 50 50 50 50 50 50

Explanation: Each element in the resource vector of platforms of class 0 is assigned a value of 50. These resources, defined in *number_ of resources*, can represent the platform's ability to engage air targets, its ability to clear minefields, etc.

Platform Range

This command sets the range of the sensors contained on each platform. Each platform has three types of notional sensors which provide information on air, sea and ground tasks, respectfully. Associated with each sensor are five ranges, each displayed by a separate ring centered on the platform. If a task is within the outer most range ring it can be seen (detected) by the platform. If the task is within the next range ring the various attributes of the task will be read (measurement), although the values may be masked/corrupted by noise. If the task is within the third range ring the task class becomes known automatically (identification). The fourth range ring denotes the range in which the task can be attacked by the platform. The last range ring indicates the range at which the platform can be attacked by the task.   By varying these ranges the scenario designer can model various abilities and limitations of the platform class.

112

Format: platform range id
        $r(1,1)$ $r(1,2)$ $r(1,3)$ $r(1,4)$ $r(1,5)$ #Air = type 1
        $r(2,1)$ $r(2,2)$ $r(2,3)$ $r(2,4)$ $r(2,5)$ #Sea = type 2
        $r(3,1)$ $r(3,2)$ $r(3,3)$ $r(3,4)$ $r(3,5)$ #Ground = type 3

id: integer number uniquely identifying the platform class.
$r(i,j)$: positive floating number.
$r(i,1)$ = detection range of platforms of class id on task type I
$r(i,2)$ = measurement range of platforms of class id on task type I
$r(i,3)$ = identification range of platforms of class id on task type I
$r(i,4)$ = attack range of platforms of class id on task type I
$r(i,5)$ = 'be-attacked' range of platforms of class id on task type I
Default $r(i,j)=(.3-.05i)*simu\_scale$

Example: platform range   0
        30.00   25.00   20.00   10.00   3.0
        20.00   15.00   10.00   5.00   2.0
        10.00   10.00   8.00   7.00   1.0

Explanation: Platforms of class 0 have been assigned values for air, sea and ground ranges. In this example, for air the detection range is set to 30, the measurement range is set to 25, and the identification range is set to 20. The range at which the platform can attack an air task is set at 10 while the range at which the platform is vulnerable to attack by an air task is set to 2. (The sea and ground ranges are different than the air.)

Platform Accuracy
This command sets the "noise" applied to the attribute values that are measured when tasks are within the platforms' measurement range. This noise allows the simulation designer to more closely model the real-world environment by introducing doubt into the attribute readings, which could be caused by a variety of elements, such as atmospheric conditions or the stealth characteristics of a target.

Format: platform accuracy id
        $e(1,1)$ $e(1,2)$ ... $e(1,natt)$
        $e(2,1)$ $e(2,2)$ ... $e(2,natt)$
        $e(3,1)$ $e(3,2)$ ... $e(3,natt)$
id: integer number uniquely identifying the platform class.
$e(i,j)$: Positive floating number which gives the standard deviation of the (Gaussian) measurement noise for sensor of type I when measuring task attribute j.
        Default: $e(i,j)=0.00$, i.e. no noise.

113

NOTE: if e(i,j) < 0.0 then that sensor·makes NO measurement of that attribute.

Example: platform accuracy 0
    0.00 0.00 0.15 0.00 0.00 0.00 0.00 0.00 0.00 -1.00
    0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 -1.00
    0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 -1.00

Explanation: In this case the sensor noise is zero for almost all attributes and sensor types. However, the third attribute of the air sensor has measurement noise with a standard deviation .15. The standard deviation of last attribute in all sensors (air, sea and ground) is set to a negative one, which will result in its not being measured by any of the sensors, of platform class 0.

Platform Subplatform

Subplatforms are previously defined platforms that reside on another platform (example: aircraft on a carrier, troops on a helicopter). Platforms of a given class may contain a number of subplatforms from different platform classes. There are several guidelines that apply to the subplatform concept:

1) The same class of subplatform (child) can reside on many platforms (parent).
2) All subplatforms of a given class are identical, independent of the platform they reside within.
3) The "depth" of the nesting structure is defined by the designer and is not limited by the software - however, loops are not allowed. (i.e.: A⊃ B⊅ A)

This command sets several specifications.

1) The number of subplatform classes that a specific platform class has.
2) The name of the subplatform classes.
3) How many of each subplatform is contained onboard the platform.
4) Who owns the various subplatforms when they are launched.

NOTE: The default is no subplatforms present on platforms

Format: platform subplatform id num
    sub_class(1)   sub_class(2) ...   sub_class(num)
    n(1)   n(2)   ...   n(num)
    owner(1)   owner(2)   ...   owner(num)
 # id: integer number, giving the unique identification of the platform class.
 # num: integer number, indicates the number of subplatform classes that this platform class has, max=7 NOTE: The maximum number of subplatforms that can be

114

'on-screen' at any one time is 100.
sub_class(I):   the name of the subplatform class. This name must be one of the class
                given in an earlier *platform general* command.

NOTE: The platform and subplatform classes must be defined (via the *platform general*
command) BEFORE the definition of their nesting structure.

n(I): the number of individual subplatforms of class 'sub_class(I)' on board this
        platform class.
owner(I): the number of the decisionmaker who will own the individual subplatforms
            of class 'sub_class(I)' when they are launched.

NOTE: If owner(I)= -1 these subplatforms will be owned by whomever "owns" the parent
platform - e.g. this gives the designer the ability to model self defense weapons.

Example: platform subplatform 15 3
           VF   VA   H60
           3   2   1
           4   0   0

Explanation: Each platform of class 15 (parent) will have three subplatform classes
(children) onboard. The three subplatform classes are VF, quantity 3, owned by DM4; VA,
quantity 2, owned by DM0; and H60, quantity 1, owned by DM0. (See previous *platform
general* command for definition of class VF.)

3.2 Task class information

Task Class Number
This command sets the total number of task classes available for the scenario.

Format: task classes ntc
   #   ntc : integer number, with the maximum 50. Default= 6

Example: task classes 24

Explanation: The total number of task classes is set to 24

Task General
This command sets the parameters for a specific task class.   These initial specifications

clearly define the task and form the basis for task uniqueness. The values of each parameter and their various combinations provide a broad range of options for the scenario designer to create models of real-world threats. These values are used each time a task of this class is created.

Format: task general id type class mv p threat_flg [icon_file]
     # id: integer uniquely identifying this task class, starts from 0.
     # type: single character A, S, G for Air, Sea or Ground tasks. Default = 'A'
     # class: string of up to 3 characters which gives a name to this class of tasks.
           Default= class#0='A', class#1='B' etc.
     # mv: maximum velocity of this class of tasks. Default=0.0040*simu_scale
     # p: appearance probability of this class of task. Default p=1/ntc, where ntc is
           total number of task classes.
     # threat_flg: indicates whether the task is a threat or a neutral. Default=1
        = 0 task is a neutral
        = 1 task is a threat to the penetration zones specified in *task penetration* command.
        = 2 task is a threat to specified penetrations zones and also to platforms, within a
            radius Ri5 (specified in *platform range*) where the platform and task are both
            of type I.
        = 3 task is a threat to specified penetrations zones and also to platforms of any type
            within a radius Ri5 (specified in the *platform range* command) where the task
            is of type I.
        = 9 task is a 'job' to do.
     # icon_file: (optional) The name of an X bitmap icon file located in ~/usr/icon

Example: task general 0 G HL 0.0 0.0424 9 hill.icon

Explanation: This statement defines the task of "taking the hill". Tasks of this class have id=0, are of type ground, and named HL. The maximum velocity is set to zero. The threat flag is 9, which indicates that this task represents a job or mission to be performed. The icon used to portray such tasks in the game will be hill.icon, which overrides the default icon file for standard ground tasks.

Task Mean

This command sets the default values for the attribute vector. Unless overridden latter in the *task attributes* portion of the scenario generator all tasks of the same task class will have these mean values assigned to their attribute vector.

Format: task mean id   a(1) a(2) ... a(natt)
     # id: integer number uniquely identifying the task class.
     # a(I):   positive floating number, mean value of the attributes I of the task class.

Default a(I)=I

Example: task mean 0   10.00 20.00 0.00 0.00 5.00 0.00 0.00 0.00 0.00 1.00

Explanation: Task Class number zero will have the above values assigned as the default values for the attribute vector.

Task Sigma
This command sets the standard deviation of the tasks' attribute vector. This allows the simulation designer to "smear" the attribute values that are associated with a specific task. Unlike the *platform accuracy* command, which affects the sensors of a specific platform class, this command affects the actual task values, independent of the platform, so the variation in attribute values is applied to all platforms and all sensors that can "see" this task class.

Format: task sigma id   sigma(1) sigma(2) ... sigma(natt)
  # id: integer number, identification of the task class.
  # sigma(I): the value of the standard deviation on attribute I. Default sigma(I)=0.00

Example: task sigma 0 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00

Explanation: Task of class 0 will have their attribute vector generated without additional values being added or subtracted to the mean attribute vector . (No sigma applied)

Task Mapping
This command specifies a linear mapping from the attributes of a task class to the resource requirements for a successful attack on <u>that class</u>. The structure of the mapping for the resource vector is: r =Aa+b, which allows for an assignment of attribute values to resource values <u>and</u> for the addition of a constant. This allows the simulation designer to control the resource match up between tasks and platforms, which enables a representation of the differences in firepower and engagibility found in the real-world.

Format: task mapping id
           A(1,1)   A(1,2)   ... A(1,natt)   B(1)
           A(2,1)   A(2,2)   ... A(2,natt)   B(2)
           A(nres,1) A(nres,2) ... A(nres,natt) B(nres)
  # id: integer number, identifying the task class.
  # A(i,j): Floating number, which indicates the mapping from task attributes to
           resources required to attack a task of this class. Default: A: ith row = 0,
           EXCEPT element A(I, I+2) = 1; if (I+2<=nattr)
  #B: all B(I) = 0

117

Example: task mapping 0
            0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00   0.00
            0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00 0.00   0.00
            0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00 0.00   0.00
            0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00 0.00   0.00
            0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00 0.00   0.00
            0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00 0.00   0.00
            0.00 0.00 0.00 0.00 0.00 0.00 0.00 0.00 1.00 0.00   1.00

Explanation: This matrix directly maps the attributes into required resources for tasks of class 0. The first line assigns resource one to attribute three, the second line assigns resource two to attribute four, etc. The 1.0 in the B vector of resource seven indicates that in addition to the one-to-one assignment from attributes to resources, one more unit of resource seven is needed.

Task View

This command determines the team information structure for the detection of a specific task class by any platform owned by a specific DM. This allows the simulation designer to tailor the tactical picture of the DM's to more closely resemble the real-world. (Various degrees of "Global awareness" can be modeled via this command)

Format: task view id v(0) v(1) v(2) ... v(ndm-1)
    # id: integer number, uniquely identifying the task class.
    # v(I): integer number which specifies the degree to which the decisionmaker Dmi
             can observe this class of tasks. Its value can be 0,1 or 2. Default v(I)=2
        = 0: DMi cannot see a task of this class unless the task happens to be within the
              detection range of at least one of his own sensors.
        = 1: DMi can see a task of this class if the task is detected by any of the sensors on
              platforms owned by other DMS.
        = 2: DMi can see a task of this class and get attribute/class information if the task
              is within the measurement/identification ranges of any of the sensors on
              platforms owned by other DMS.

Example: task view 0 2 2 2 2 2 2

Explanation: Task Class number zero can be seen by all Decisionmakers. Attribute values of the task can be seen by all DM's if it is within the sensor range of any platform owned by any decisionmaker.

Task Stealth

This optional command resets the detection, measurement, identification, attack and be-

118

attack ranges for a specific task class vis-a-vis a specific platform class. (Overrides default values.) This allows the scenario designer to tailor ranges down to an individual threat or mission, which can be used to create the possibility of a late detection or non-detection of a unique threat. (Example: ground or sea mines)

Format: task stealth id1 id2 [sensor(1) sensor(2) sensor(3) attack(1) attack(2)]
    # id1:   integer number, identifying the task class.
    # id2:   integer number, identifying the platform class.
    # sensor(I): floating point numbers:
        I=1: overrides sensor detection range $r(j,1)$ of platforms of class id2 when encountering tasks of class id1 (that are of type j)
        I=2: overrides sensor measurement range $r(j,2)$ of platforms of class id2 when encountering tasks of class id1 (that are of type j)
        I=3: overrides sensor identification range $r(j,3)$ of platforms of class id2 when encountering tasks of class id1 (that are of type j)

Default: values $r(j,i)$ as specified in *platform range* command -- EXCEPT if threat_flg = 2 or 3, in which case default values = $r(j,5)$

    # attack(I): floating point numbers:
        I=1: overrides associated attack range $r(j,4)$ of platforms of class id2 when attacking tasks of class id1 (that are of type j). Default: value = $r(j,4)$, EXCEPT if threat_flg = 2, in which case default value = 0.0
        I=2: overrides associated be-attacked range $r(j,5)$ of platforms of class id2 when encountering tasks of class id1 (that are of type j). Default: value = $r(j,5)$ for all values of threat flag.

Example: task stealth 0 0 100.00   90.00   80.00   20.00   0.00
           task stealth 0 1   40.00   30.00   20.00   5.00   2.00

Explanation: Task class 0 has been assigned stealth values vis-a-vis platform class zero and one.   In this example, if we assume that task class 0 is of type air, then the new ranges will apply to the Air ranges of the platforms specified. Line one sets the ranges for task/platform 0/1 pairings at 100 for detection, 90 for measurement, 80 for identification, 20 for attack and 0 for be-attack, which will prevent the task from being able to attack the platform. These new values will override the default value of the Air range for platforms zero and one, previously specified *platform range* against this task class.   All other ranges, Sea and Ground, will retain the default values specified in *platform range* command.

Task Expertise (Not used in the current version of DDD - default values are hard-coded.)

119

This command sets the level of expertise which the decisionmaker has to interpret the task attribute measurements. This allows the scenario designer to "level the playing field"; which has a variety of uses, such as adjusting for the varying professional backgrounds of the decisionmakers, which may affect team performance.

Format: task observe expertise id
            O(0,1)   O(0,2)   ... O(0,natt-1)
            O(1,1)   O(1,2)   ... O(1,natt-1)
            O(ndm-1,1) O(ndm-1,natt-1) ... O(ndm-1,natt-1)
     # id: integer number, specifying the task class.
     # O(i,j): floating number to indicate the relative observation expertise for DMi on
            attribute j for the tasks of the specified class, O(i,j) must be from 0.1 to 1,
            Default O(i,j) = 1.0

Example: Task observe expertise 1
            1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
            1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
            1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
            1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
            1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0
            1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0 1.0

Explanation: In this example all DM's have the same abilities to observe all task attributes.

Task Destroy (Not used in the current version of DDD - default values are hard coded.)
This command sets the ability of the decisionmaker to destroy the specific task class. Like *task expertise*, this command allows the user to "level the playing field" if the professional backgrounds of the decisionmakers is such that a wide diversity of playing skills is expected.

Format: task destroy expertise id d(0) d(1) d(2) ... d(ndm-1)
     # id: integer number, identifying the task class.
     # d(I): floating number which indicates the degree of expertise DMi has in attacking
            this class of tasks. It can take on value from 0 to 1, Default d(I)=1.0

Example: task destroy expertise 1   1 1 1 1 1 .5

Explanation: In this example, all Dm's have the same ability to destroy task number 1 with the exception of DM5, who has a "handicap of 50% in his ability to destroy the task.

120

Task Attack

This command sets the ability of various decisionmakers to attack tasks of a given class. This allows the scenario designer to control who can attack a given task class, which can be used to highlight command structure differences as well as to control competition for assets needed for the attack(s)..

Format: task attack id DMresp at(0) at(1) at(2)... at(ndm-1)
    # id: integer number, specifying the task class.
    # DMresp: DM responsible for this class of tasks. Default=0
    # at(I): integer flag which indicates whether DMi is (or can be) assigned to attack tasks
        of this class. Default at(I)=1
    = 0, DMi is not assigned to attack the tasks.
    = 1, DMi is assigned to attack the tasks.

Example: task attack 1 2 1 1 1 1 1 1

Explanation: In this example, the responsibility for tasks of class 1 is assigned to DM2, but all DM's have the ability to attack tasks of this class when identified.

Task Penetration

This command sets the sensitivity of the penetration zones defined earlier in *penetration zones* to the various task classes. This allows the scenario designer to create areas that must be defended from attack by specific task classes.

Format: task penetration id p(0) p(1) ...p(npen-1)
    # id   : The task class to which this command refers.
    # npen: The number of penetration zones defined by *penetration number* earlier,
    # p(I) : = 1 means class#id is capable of doing damage to penetration zone#I.
    #   = 0 means class#id cannot do damage to penetration zone#I.
        Default p(I)=1
Example: task penetration 0 0 0 0 0 0 0 0
        task penetration 1 0 0 0 0 1 1 0

Explanation: In line one the task class 0 cannot do damage to any of the seven penetration zones defined. In line two the task number 1 can do damage to penetration zones four and five.

4. STATE INFORMATION

The commands listed in this section provide the user the ability to specify parameters

121

of *individual* platforms and tasks. If no specific state information is provided the scenario generator will create values derived from the values contained in the previous class commands.

Platform Number
This command specifies the total number of individual platforms used in the simulation.

Format: platform number nplat
    # nplat : integer number, with the maximum 99.   Default 6

Example: platform number 11

Explanation: There are a total of eleven platforms in this simulation.

Platform Definition
This command allows the user to specify each individual platform used in the simulation, as well as which decisionmaker has initial control over the platform and where it is located at clock start. This command allows the scenario designer to specify order of battle for own forces.

Format: platform definition id class dm x y
    # id: integer number between 0 to 100, giving a unique platform id.
    # class: string of up to 3 chars giving the unique name of the class this platform
             belongs to as described in 'platform general' command, Default=name of class
             #0 as described in 'platform general' command.
    # dm: the DM this platform belongs to initially. Default=1
    # x: x coordinate of the platform initial position.
    # y: y coordinate of the platform initial position.
             Default (x, y) coordinates are generated randomly
Example: platform definition 0 CVN 4 70.00 15.00

Explanation: This places platform number 000 at xy grid location (70,15). It is a CVN, and is owned initially by DM4.

NOTE: It is important that the designer provide a definition for the total number of platforms specified in *platform number,* otherwise the scenario generator will create default platforms (of class 0) to make up the total number.

Task Definition
This command allows the user to specify class and initial responsibility for each individually numbered task to be used in the simulation, as well as the tasks' priority to the

decisionmaker who is responsible for this task. This allows the scenario designer to specify class and responsibility for each individual task, uniquely identifying each "target" or "mission" presented to the team.

Format: task definition id classnum iDM priority
    # id: integer number between 200 and 399 giving a unique id to the task.
    # classnum: the class number to which this task belongs. Default: generated per
              probabilities given in 'task general' command
    # iDM: the DM who is responsible for this task, Default=0
    # priority: priority of this task (an integer 0,1,2, or 3), Default=0

Example: task definition 200 21 1 0

Explanation: Task number 200, task class 21, is the responsibility of decisionmaker one, and has a (initial) priority of zero.

Task Attributes
This command can set the attribute values for an individual task specified in the simulation. This allows the scenario designer to specify each task uniquely.

Format: task attributes id
      at(1)   at(2) ... at(natt)
    # id: integer number from 200 to 399, giving an unique id to the task.
    # at(I): positive floating number, attributes of this task, Defaults are generated
          randomly by task class according to values described in the *task mean* and *task sigma* commands.

Example: task attributes 200
      02.0 10.0   00.0 00.0 20.0 00.0 00.0 00.0 00.0 00.0

Explanation: In this example the task attributes are set for task number 200, overriding the initially defined default values. Each value above will be assigned to the appropriate position in the attribute vector.

Task Resource
This command sets the amount of resources required to properly engage the task (and receive full point values - engaging task with less than the required amount will result in "partial credit") This overrides any default values previously defined. This allows the scenario designer to further tailor the properties of individual tasks to platforms.

Format: task resource id

123

        res(1) res(2) ... res(nattr)
  # id: integer number from 200 to 399, giving a unique id to the task.
  # res(I): positive floating number, resources required to prosecute the task. Defaults
are generated from task attributes according to the procedure described in *task*
        mapping.

Example: task resources 200
        0 0 20 0 0 0 0

Explanation: In this example task 200 will require platform(s) with resources of at least $res(3)=20$ in order for it to be properly engaged.

Task Prerequisites

This command sets an "order" in which selected tasks must be prosecuted. This allows the scenario designer to cause the team to plan courses of action and mission priorities.

Format: task prerequisites id num pr(1) pr(2) ... pr(num)
  # id: integer number from 200 to 399, giving a unique id to the task.
  # num: integer number of prerequisite tasks that this task has
  # pr(I): the task number of a prerequisite for this class

NOTE: prerequisites must have lower task number to avoid cycles [i.e. $pr(I) < id$]
       Default=task has no prerequisites (num=0)

Example: task prerequisites 310 3 236 237 238

Explanation: In this example task 310 has three prerequisites that must be accomplished prior to engagement. If tasks 236,237, and 238 are not completed or have not disappeared (not necessarily in any particular order) the simulation will not allow an attack on task 310.

Task Bias

This command is used to introduce a bias, or offset, in the measurements of a given tasks' attributes. The bias will be reduced to 0 linearly when the task is within attack range of the platform whose sensors are measuring the attributes.

Format: task bias id
      b(1) b(2) ... b(nattr)
  # id: integer number from 200 to 399, giving an unique id to the task.
  # b(I): floating point number, bias in task #id when reading attribute #I.
       Default b(I)=0.00

Example: task bias 211
                0.0 0.0 1.0 1.0 -2.0 10.0 1.0 0.0 0.0 0.0

Explanation: Task 211, when measured, will have a bias applied, with the above values added or subtracted to its attribute vector.

Task Spawn
This commands sets a parent/child relationship for tasks. As the "parent" task is either engaged or destroyed a "child" task will then appear. This allows the scenario designer to include a measure of mission "growth" in his initial planning of the scenario.

Format: task spawn id num type sid(1) ... sid(5)
                          sid(6) ... sid(10)
                          sid(num-4) ... sid(num)
    # id: integer id of 'spawner' task for which new tasks will be spawned.
    # num: integer number of tasks to be spawned.
    # type: character A or D denoting a task Attack or task Disappear event.
    # sid(I): integer ids of 'spawned' tasks.

    NOTE: I) 0 < num < n, where n is specified via *number_of tasks* command.
              ii) a task can be both spawned and a spawner (ie, recursive),
                  however, a task cannot be spawned by more than one spawner.
              iii) to prevent spawning cycles, ie, X spawns Y spawns X, we require id
                   < sid(I), I = 1..num.
              iv) at most 5 spawned task ids specified per line, and, for num > 5, only
                  last line can have <= 5 task ids specified.

Example: task spawn 286 1 A 289

Explanation: In this example task 286, when attacked, will spawn task 289

Task Remove
This command inhibits arrival of future tasks depending on either the attack or disappearance of a given task. This can be used by the scenario designer to "anticipate" paths that may or may not be followed by the DM's during the simulation.

Format: task remove id num type did(1) ... did(5)
                          did(6) ... did(10)
                          did(num-4) ... did(num)

125

id: integer id of 'remover' task for which future tasks will be removed.
num: integer number of tasks to be removed.
type: character A or D denoting a task Attack or task Disappear event.
did(I): integer ids of 'removed' tasks.

NOTE: I)   0 < num < n, where n is specified via *number_of tasks* command.
          ii) a task can be both removed and a remover (ie, recursive), moreover,
               a task can be removed by more than one remover.
          iii) at most 5 removed task ids specified per line, and, for num > 5, only
               last line can have <= 5 task ids specified.

Example: task remove 211 4 A   234   235   236   256

Explanation: In this example the attacking of task 211 will prevent 4 new tasks from
entering the scenario. This particular example was an artillery unit, in which by attacking
the launcher no further rounds will be fired at friendly units.

Game End
This command allows the scenario designer to specify the mission completion point, in
terms of attacking a specific task, which may occur before the time specified in *game
duration.*

Format: game end id time
          # id: game will gracefully end after the task number id has been attacked.
          # time: float delay time to wait before ending the game. Default: id=399, time=5.0.

Example: game end 390 5.00

Explanation: In this example the scenario run will terminate 5 seconds after an attack on
task 390 completes.

5. MANEUVER INFORMATION

    Maneuvers describe the track of a task, which includes turning points, velocity and the
"life span". For each task the scenario generator allows the user to enter up to five
maneuver segments. The first element of the maneuver command defines the starting point
for the task, the last element defines the ending point. The commands between the start
and end point fall into two categories - ordinary and staying maneuvers. An ordinary
maneuver moves the task between a starting point and a destination point at the designated
velocity. A staying maneuver designates a staying period at the given position. The

126

velocity used in this command is a relative velocity with a range from 0 (no velocity) to 1 (maximum velocity as specified in task class information).

Format: maneuver definition id time
         flag(1) x(1) y(1) v(1)
         flag(2) x(2) y(2) v(2)
         flag(m) x(m) y(m) v(m)
    # id: integer number uniquely specifying this task.
    # time: floating number specifying the task arrival time. If this number is omitted, a randomly generated arrival time will be assigned to the task.
    # flag(I): one character to indicate the type of the maneuver.
         # =m   ordinary maneuver
         # =s   maneuver of staying at a point
         # =e   ending maneuver
    # v(I):   relative velocity (0<vi<=1)., If flagi=s, then vi is the task's staying time rather than its velocity.
    # x(I),y(I): coordinates .
    Default: straight line maneuvers automatically generated, starting on a circle of radius = (simulation scale)/2 and ending in the center of the screen.

Example: maneuver definition   200 1.00
            s 18.00 89.00 3600.00
            e 18.00 89.00 0.00

Explanation: In this example task 200 will appear at time 1.0 on the scenario clock. It will appear at XY grid coordinates (18, 89) and remain stationary for a period of 3600 seconds, then disappear.

Example: maneuver definition   21 60.00
            m 00.00 00.00 0.90
            m 30.00   40.00 0.80
            e 60.00 60.00 0.00

Explanation: In this example task 201 will appear at time 60.0 on the scenario clock. It will appear at XY grid coordinates (00,00) and proceed to grid coordinates (30,40) with a speed of .9 times the default maximum speed of the task class (defined in *task definition*). Upon reaching the coordinates (30,40) the task will slow to .8 of maximum and turn towards the final grid coordinates of (60,60), where upon reaching them the task will disappeared,

6. SETTING UP AN EXPERIMENTAL SPECIFICATION FILE

The procedure for generating the required files following scenario development is rather straight forward. In order to understand how the scenario generator turns the designed parameters into a scenario the file structure must be review.

6.1 Scenario files
There are two scenario data files that are used to set up the experimental environment: the experiment specification file (*xs*) and the listing file (*ls*).

6.1.1 Experiment specification file (*xs*)
The *xs* file is the input file to the scenario generator, which contains all designer specified commands and values reviewed in the above sections. The file name is *xsnnnn.dat*, where "*xs*" stands for experiment specification and the *nnnn* represents a four digit string used to identify the specific scenario. Each *xs* file contains information for one scenario - for each additional scenario multiple *xsnnnn.dat* files are required. The *xsnnnn.dat* file is used to describe the scenario parameters to the scenario generator. The file is written in experimental specification language (xsl) developed to provide users a friendly interface. There are several very basic restrictions regarding the writing of the xs file:

- Blank lines are not allowed - use # as a separator between lines

- Command keywords must be a string of lower case characters

- The end of the class information block must include a flag: *end_of_class_info* The flag is used by the scenario generator as a control signal.

- Vectors and matrixes dimensions and sizes must match the limits specified either by the designer or the default value. Missing or extra data will cause generator errors.

- The order of the commands is important. The precedence outlined in the previous sections must be followed.

- Comments may be used, however, the line must start with a #, which makes the entire line a comment line.

6.1.2 Listing file (*ls*)
The *ls* file is the output file of the scenario generator, which is used in the controller window to run the simulation. This file contains not only the information provided by the

designer in the *xs* file but also any default and "filler" information needed to run the simulation. The scenario generator uses the default values of all commands and data strings to create the needed values. The file name is *lsnnnn.dat*, where *ls* stands for listing and the *nnnn* represents the same four digit string used in the *xs* file definition.

6.2 scenario generation procedures
After the scenario designer has determined the parameters of the desired simulation an *xs* must be created that contains all the relevant information. A "blank" *xs* file, or template, is edited to fill in these desired parameters.

Once completed, the *xs* file is then run through the scenario generator using the following command:

sg xsnnnn (using the appropriate file number)

If all information contained in the *xs* file has been entered in accordance with this guides previous sections an *ls* file will be generated. This *ls* file can then be used by the controller window to run the scenario.

If any errors in the format of the *xs* exist the scenario generator will respond with an error message indicating that errors exist and in what line of the *xs* file they can be found.

7. EXAMPLE FILE

```
#<<<<<<<<<<<<<<<< General Items >>>>>>>>>>>>>>
#
 decision structure 6
         CJTF  GCC  MEU1  MEU2  CVBG   ARG
         0      0    1    1    0     0
#---------
simulation time 2400.0
#---------
number_of tasks 140
#---------
simulation scale 100.0
#---------
number_of attributes 10
     Value  Time  Air  Sea  Ground  hold  Mine  tank  Med   enemy
#---------
number_of resources 7
 air sea ground hold mine tank Med
#---------
renew interval 0.5
#---------
random seed  1
#---------
message number 4
#--------
 comm epoch 6.0
#---------
comm delay 5.0
#---------
request delay 5.0
#---------
transfer delay 10.0
#---------
attack delay 15.0
#---------
scramble time 0
#---------
scramble id 0
#---------
 communication permission
   0   1   1   1   1   1
   1   0   1   1   1   1
   0   1   0   1   1   1
   0   1   1   0   1   1
   1   1   1   1   0   1
   1   1   1   1   1   0
#---------
land area 0.00 40.0 30.0 100.0
#---------
```

draw line 5.0 95.0 5.0 75.0 2
draw line 5.0 75.0 19.5 83.0 2
draw line 19.5 83.0 26.5 83.0 2
draw line 15.0 68.0 26.6 72.0 2
draw line 15.0 50.0 15.0 68.0 2
draw line 15.0 50.0 25.0 45.0 2
#---------
draw rectangle 27.0 49.5 2.5 4.0
draw rectangle 27.0 64.5 2.5 4.0
draw rectangle 27.0 87.0 2.5 4.0
#---------
penetration number 7
#---------
penetration zone circle   0 25.00 45.00 5.00   # port area
penetration zone circle   1 24.00 60.00 5.00   # high ground area
penetration zone rectangle 2 26.5   70.00 4.00 6.0 # beach area one
penetration zone rectangle 3 26.5   80.00 4.00 6.0 # beach area two
penetration zone circle   4 5.00   95.00 5.00   # Airfield
penetration zone circle   5 70.00 15.00 7.00   # fleet area (CVN)
penetration zone circle   6 64.00 75.00 4.50   # fleet area (amphib)
#
#<<<<<<<<<<<<<<<<< Platform Class Information >>>>>>>>>>>>>>>
platform classes 26
#---------
platform 0: test platform for overall coverage
platform general 0 A   GOD 0.99 1
            1 1 3000.000 3.000 20.000
platform resource 0 50 50 50 50 50 50 50
platform range   0
 100.00 99.00 98.00 97.00 00.00
 100.00 99.00 98.00 97.00 00.00
 100.00 99.00 98.00 97.00 00.00
platform accuracy 0
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
#
platform 1: carrier based fighter aircraft
platform general 1 A   VF   0.45 1
            0 0 600.000 3.000 10.000
platform resource 1   5 2 0 0 0 0 0
platform range   1
  15.0 14.00 13.00 10.00 3.00
  15.0 14.00 13.00 10.00 3.00
  0.0 0.00   0.00   0.00 0.00
platform accuracy 1
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 -1.000
#
platform 2: shore based attack fighters

platform general 2 A F15 0.45 1
       1 1 2700.00 3.00   300.00
platform resource 2 5 1 1 0 0 0 0
platform range 2
 15.00   14.00   13.00   10.00 3.00
 15.00   14.00   13.00   10.00 3.00
  0.00   0.00   0.00   0.00 0.00
platform accuracy 2
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 -1.000
#
platform 3: carrier based attack aircraft
platform general 3 A   VA 0.25 1
       1 1 2700.000 3.000 10.000
platform resource 3   0 5 5 0 0 5 0
platform range   3
 15.00 14.00 13.00   5.00 3.00
 15.00 14.00 13.00 10.00 3.00
 15.00 12.00 11.00 10.00 3.00
platform accuracy 3
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 -1.000
#
platform 4: carrier based MCM helicopter
platform general 4 A   MCM   0.40 1
       1 1 2700.00 3.000 10.000
platform resource 4   0 0 0 0 2 0 0
platform range   4
12.00 10.00 9.00   5.00 3.00
12.00 10.00 9.00   5.00 3.00
12.00 10.00 9.00   5.00 3.00
platform accuracy 4
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 -1.000
#
platform 5: SH60 carrier/frigate helicopter
platform general 5 A   H60 0.40 1
       1 1 900.00 3.000 20.000
platform resource 5   0 6 0 0 0 0 0
platform range 5
12.00 10.00 8.00   5.00 3.00
20.00 10.00 8.00   5.00 3.00
10.00   9.00 8.00   5.00 0.00
platform accuracy 5
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 -1.000
#

platform 6: amphib based attack helicopter
platform general 6 A   HCB 0.40 1
              1 1 2700.000 3.000 10.000
platform resource 6   0 0 5   0 0 10 0
platform range   6
 12.00 11.00 10.00 5.00 3.00
 12.00 11.00 10.00 5.00 3.00
 12.00 11.00 10.00 8.00 3.00
 platform accuracy   6
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 0.000
#
platform 7: amphib based troop carrier helicopter
platform general 7 A   HTP 0.45 1
              1 1 2700.000 3.000 10.000
platform resource 7   0 0 5 5 0 0 0
platform range   7
 12.00 10.00 9.00   5.00 3.00
 12.00 10.00 9.00   5.00 3.00
 10.00 9.00 8.00   5.00 3.00
 platform accuracy   7
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 -1.000
#
platform 8: amphib based Medivac helicopter
platform general 8 A   HMV 0.30 1
              1 1 2700.000 3.000 15.00
platform resource 8   0 0 0 0 0 0 5
platform range   8
 12.00 10.00 8.00   5.00   3.00
 12.00 10.00 8.00   5.00   3.00
 10.00 9.00 8.00   5.00   3.00
 platform accuracy   8
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 -1.000
#
platform 9: amphib based engineering company helicopter
platform general 9 A   HE 0.40 1
              1 1 2700.00 3.000 20.000
platform resource 9   0 0 0 0 5 0 0
platform range   9
 12.00 10.00 8.00   5.00 3.00
 12.00 10.00 8.00   5.00 3.00
 10.00 9.00 8.00   5.00 0.00
platform accuracy   9
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000   0.000 0.000 0.000 -1.000

133

```
#
# platform 10: recon aircraft
platform general 10  A   SR7 0.50  1
            1 1 1200.000 3.000 30.000
platform resource 10   0 0 0 0 0 0 0
platform range   10
 25.00  20.00  15.00    0.00   0.01
 50.00  50.00  45.00    0.00   0.01
 50.00  50.00  45.00    0.00   0.01
platform accuracy  10
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 0.000
#
# platform 11: Sea born SAM
platform general 11  A   SAM 0.99 0
            0 0 60.00 3.000 10.000
platform resource 11   5 1 0 0 0 0 0
platform  range  11
 40.0    0.00    0.00   40.00 0.00
  0.00   0.00    0.00    0.00 0.00
  0.00   0.00    0.00    0.00 0.00
 platform accuracy   11
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
#
# platform 12: ship based NGFS support rounds
platform general 12  A   5I 0.00  0
            0 0 60.00 3.000 10.000
platform resource 12   1 1 5 0 0 2 0
platform range   12
 0.00 0.00 0.00 0.00 0.00
 0.00 0.00 0.00 0.00 0.00
45.00 0.00 0.00 40.00 0.00
platform accuracy   12
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
#
# platform 13: stinger detachment
platform general 13  a   SD 0.4  1
            1 1 2700.000 3.000 10.00
platform resource 13   6 0 0 0 0 0 0
platform range   13
 8.00   8.00  8.00  8.00  0.00
 0.00   0.00  0.00  0.00  0.00
 0.00   0.00  0.00  0.00  0.00
platform accuracy  13
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
```

```
0.000  0.000  0.000  0.000  0.000  0.000   0.000  0.000  0.000  -1.000
#
# platform 14: Frigate
platform general 14  S   FFG  0.20  1
          1 1  2700.000  3.000  20.000
platform resource 14   1  5  0  0  1  0  0
platform range   14
 35.0   15.00    7.00   6.00  2.00
 35.0   15.00    7.00   6.00  2.00
  0.00   0.00    0.00   0.00  0.00
platform accuracy  14
0.000  0.000  0.000  0.000  0.000  0.000   0.000  0.000  0.000  -1.000
0.000  0.000  0.000  0.000  0.000  0.000   0.000  0.000  0.000  -1.000
0.000  0.000  0.000  0.000  0.000  0.000   0.000  0.000  0.000  -1.000
#
# platform 15: aircraft carrier
platform general 15  S   CVN  0.0  0
          1 1  2700.000  3.000  20.000
platform resource 15   1  1  0  0  1  0  0
platform range   15
 100.0  20.00  20.00  2.00  3.00
 100.0  20.00  20.00  2.00  3.00
   0.00   0.00   0.00  0.00  0.00
platform accuracy  15
0.000  0.000  0.000  0.000  0.000  0.000   0.000  0.000  0.000  -1.000
0.000  0.000  0.000  0.000  0.000  0.000   0.000  0.000  0.000  -1.000
0.000  0.000  0.000  0.000  0.000  0.000   0.000  0.000  0.000  -1.000
#
# platform 16: Cruiser
platform general 16  S   CG  0.2  1
          1 1  2700.000  3.000  20.000
platform resource 16   1  1  0  0  1  0  0
platform range   16
 40.0   20.00  20.00  00.00  3.00
 15.00  14.00  12.00  00.00  3.00
  0.00   0.00   0.00   0.00  0.00
platform accuracy  16
0.000  0.000  0.000  0.000  0.000  0.000   0.000  0.000  0.000  -1.000
0.000  0.000  0.000  0.000  0.000  0.000   0.000  0.000  0.000  -1.000
0.000  0.000  0.000  0.000  0.000  0.000   0.000  0.000  0.000  -1.000
#
# platform 17: Destroyer one
platform general 17  S   DD1  0.0  0
          1 1  2700.000  3.000  20.000
platform resource 17   1  1  0  0  1  0  0
platform range   17
 40.0   20.00  20.00  00.00  3.00
 15.00  14.00  12.00  00.00  3.00
  0.00   0.00   0.00   0.00  0.00
platform accuracy  17
0.000  0.000  0.000  0.000  0.000  0.000   0.000  0.000  0.000  -1.000
```

135

```
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
#
# platform 18: Destroyer two
platform general 18  S   DD2  0.0  0
              1 1 2700.000  3.000  20.000
platform resource 18    1 1 0 0 1 0 0
platform range   18
  40.0  20.00  20.00  00.00  3.00
  15.00  14.00  12.00  00.00  3.00
   0.00   0.00   0.00   0.00  0.00
 platform accuracy  18
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
#
# platform 19: Amphibious assault ship (MEU 1 embarked)
platform general 19  S   LA  0.00  0
              1 1 2700.000  3.000  20.000
platform resource 19    1 1 0 0 1 0 0
platform range   19
  20.00  20.00  20.00  00.00  3.00
  15.00  14.00  12.00  00.00  3.00
   0.00   0.00   0.00  00.00  0.00
platform accuracy   19
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
#
# platform 20: Amphibious assault ship (MEU 2 embarked)
platform general 20  S   LP  0.00  0
              1 1 2700.000  3.000  20.000
platform resource 20    1 1 0 0 1 0 0
platform range   20
  20.00  20.00  20.00  00.00  3.00
  15.00  14.00  12.00  00.00  3.00
   0.00   0.00   0.00  00.00  0.00
platform accuracy   20
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
#
# platform 21: Amphibious assault ship (CJTF assets embarked)
platform general 21  S   LH  0.00  0
              1 1 2700.000  3.000  20.000
platform resource 21    1 1 0 0 1 0 0
platform range   21
  20.00  20.00  20.00  00.00  3.00
  15.00  14.00  12.00  00.00  3.00
   0.00   0.00   0.00  00.00  0.00
platform accuracy   21
```

```
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
#
# platform 22: LCAC
platform general 22  S  LC1  0.25  0
          1 1 2700.000 3.000 10.000
platform resource 22   0 0 0 0 0 0 0
platform range  22
 0.00   0.00   0.00    0.00 0.00
 0.00   0.00   0.00    0.00 0.00
 50.00  0.00  10.00   0.00 3.00
platform accuracy  22
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
#
# platform 23: LCAC
platform general 23  S  LC2  0.25  0
          1 1 2700.000 3.000 10.000
platform resource 23   0 0 0 0 0 0 0
platform range  23
 0.00   0.00   0.0    0.00 0.00
 0.00   0.00   0.00    0.00 0.00
 50.00  0.00  10.00    0.00 3.00
platform accuracy  23
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
#
# platform 24: ground born tracked infantry company
platform general 24  G   TG  0.09 1
          1 1 2700.000 3.000 10.00
platform resource 24   0 0 5 5 0 1 0
platform range  24
 0.00   0.00   0.00    0.00 0.00
 0.00   0.00   0.00    0.00 0.00
 50.00 30.00  10.00    5.00 3.00
platform accuracy  24
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
#
# platform 25: sigonella
platform general 25  S   BAS  0.00  0
          1 1 2700.00 3.000 10.000
platform resource 25   0 0 0 0 0 0 0
platform range  25
 0.00   0.00   0.00    0.00   0.0
 0.00   0.00   0.00    0.00   0.0
 0.00   0.00   0.00    0.00   0.0
```

```
platform accuracy   25
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
0.000 0.000 0.000 0.000 0.000 0.000  0.000 0.000 0.000 -1.000
#---------
#  aircraft carrier carrying: fighter, CAS, anti-ship helo's
 platform subplatform  15 3
     VF   VA   H60
     3    2   1
     4    0   0
#
# cruiser carrying: SAM missiles
platform subplatform  16  1
     SAM
     10
     -1
#
#  Destroyer 1 (MEU 1 asset) carrying 5 Inch fire mission
platform subplatform  17   1
     5I
     10
     2
#
#  Destroyer 2 (MEU 2 asset) carrying 5 Inch fire mission
platform subplatform  18  1
     5I
     10
     3
#
#  amphibious ship (MEU1) carrying: AAAV and attack/troop/engineer helicopter
platform subplatform  19  3
     HCB  HTP  HE
     1   1   1
     2   2   2
#
#  amphibious ship (MEU2) carrying: troop/medivac/engineer helocopter
platform subplatform  20  3
     HTP   SD   HMV
     1    1    1
     3    5    3
#
#  amphibious ship (CJTF) carrying : mine-countermeasure helicopter/troop helo
platform subplatform  21 2
     MCM  HTP
     1   1
     0   0
#
# LCAC1 carrying: ground based infantry
 platform subplatform  22  1
     TG
     1
```

```
        2
#
# LCAC2 carrying: ground based infantry
 platform subplatform  23  1
        TG
        1
        3
#
#  sigonella base providing: troop helos
platform subplatform  25   2
      F15  SR7
       1   1
       0   0
#SP TK
#<<<<<<<<<<<<<<<< Task Class Information >>>>>>>>>>>>>>>
task classes 24
#---------
# task 0: ground mission (hill)
task general 0  G  HL  0.10  0.211  9 hill.icon
task mean  0  10.0  20.0  0.00  0.00  5.00  0.00  0.00  0.00  0.00  1.0
task view  0 2  2  2  2  2  2
task stealth  0   0 100.0 100.0 100.0 00.0 0.0
task stealth  0 14 100.0 100.0 100.0 00.0 0.0
task stealth  0 15 100.0 100.0 100.0 00.0 0.0
task stealth  0 16 100.0 100.0 100.0 00.0 0.0
task stealth  0 17 100.0 100.0 100.0 00.0 0.0
task stealth  0 18 100.0 100.0 100.0 00.0 0.0
task stealth  0 19 100.0 100.0 100.0 00.0 0.0
task stealth  0 20 100.0 100.0 100.0 00.0 0.0
task stealth  0 21 100.0 100.0 100.0 00.0 0.0
task stealth  0 22 100.0 100.0 100.0 00.0 0.0
task stealth  0 23 100.0 100.0 100.0 00.0 0.0
task attack  0  1  1 1 1 1 1 1
#
# task 1: ground mission (airport)
task general 1  G  AP  0.00  0.211  9 dullas.icon
task mean  1  30.0  10.0  0.00  0.00  10.00  10.00  0.00  0.00 0.00  1.0
task view  1 2  2  2  2  2  2
task stealth  1   0 100.0 100.0 100.0 00.0 0.0
task stealth  1 14 100.0 100.0 100.0 00.0 0.0
task stealth  1 15 100.0 100.0 100.0 00.0 0.0
task stealth  1 16 100.0 100.0 100.0 00.0 0.0
task stealth  1 17 100.0 100.0 100.0 00.0 0.0
task stealth  1 18 100.0 100.0 100.0 00.0 0.0
task stealth  1 19 100.0 100.0 100.0 00.0 0.0
task stealth  1 20 100.0 100.0 100.0 00.0 0.0
task stealth  1 21 100.0 100.0 100.0 00.0 0.0
task stealth  1 22 100.0 100.0 100.0 00.0 0.0
task stealth  1 23 100.0 100.0 100.0 00.0 0.0
task attack  1  1  1 1 1 1 1 1
#
```

139

```
# task 2: ground mission (seaport)
task general 2  G  SP  0.00  0.211  9 norfolk.icon
task mean 2  30.0  10.0  0.00  0.00  10.00  10.00  0.00  0.00  0.00 1.00
task view 2 2  2  2  2  2  2
task stealth  2  0 100.0 100.0 100.0 00.0 0.0
task stealth  2 14 100.0 100.0 100.0 00.0 0.0
task stealth  2 15 100.0 100.0 100.0 00.0 0.0
task stealth  2 16 100.0 100.0 100.0 00.0 0.0
task stealth  2 17 100.0 100.0 100.0 00.0 0.0
task stealth  2 18 100.0 100.0 100.0 00.0 0.0
task stealth  2 19 100.0 100.0 100.0 00.0 0.0
task stealth  2 20 100.0 100.0 100.0 00.0 0.0
task stealth  2 21 100.0 100.0 100.0 00.0 0.0
task stealth  2 22 100.0 100.0 100.0 00.0 0.0
task stealth  2 23 100.0 100.0 100.0 00.0 0.0
task attack  2  1  1 1 1 1 1 1
#
# task 3: ground mission (Holding/occupying)
task general 3  G  HD  0.00 0.211  9 hold.icon
task mean  3  0.0  10.0  0.00  0.00  0.00  5.00  0.00  0.00  0.00 1.00
task view  3  2  2  2 2 2 2
task stealth  3  0 100.0 100.0 100.0 00.0 0.0
task stealth  3 14 100.0 100.0 100.0 00.0 0.0
task stealth  3 15 100.0 100.0 100.0 00.0 0.0
task stealth  3 16 100.0 100.0 100.0 00.0 0.0
task stealth  3 17 100.0 100.0 100.0 00.0 0.0
task stealth  3 18 100.0 100.0 100.0 00.0 0.0
task stealth  3 19 100.0 100.0 100.0 00.0 0.0
task stealth  3 20 100.0 100.0 100.0 00.0 0.0
task stealth  3 21 100.0 100.0 100.0 00.0 0.0
task stealth  3 22 100.0 100.0 100.0 00.0 0.0
task stealth  3 23 100.0 100.0 100.0 00.0 0.0
task attack  3  1  1 1 1 1 0 0
#
# task 4: ground mission (Taking)
task general 4  G  TK  .00 0.211  9 take.icon
task mean  4  10.0  10.0  0.00  0.00  5.00  0.00  0.00  0.00  0.00 1.00
task view  4  2  2  2 2 2 2
task attack  4  1  1 1 1 1 0 0
#
# task 5: ground contact (artillery)
task general 5  G  AT  .99 0.211  2  artillery.icon
task mean  5  2.0  10.0  0.00  0.00  0.00  0.00  0.00  2.00  0.00 1.00
task view  5 2  2  2  2  2  2
task stealth  5  0 100.0 100.0 100.0 100.0 00.0
task stealth  5  1 00.0  00.0  00.0  5.0 00.0
task stealth  5  2 00.0  00.0  00.0  5.0 00.0
task stealth  5  3 00.0  00.0  00.0  8.0 00.0
task stealth  5  4 00.0  00.0  00.0  00.0 00.0
task stealth  5  5 00.0 00.0  00.0  00.0 00.0
task stealth  5  6 00.0  00.0  00.0  6.0 00.0
```

140

```
task stealth  5  7  00.0  00.0  00.0   5.0 00.0
task stealth  5  8  00.0  00.0  00.0  00.0 00.0
task stealth  5  9  00.0  00.0  00.0  00.0 00.0
task stealth  5 10 100.0  50.0  50.0  00.0 00.0
task stealth  5 11  00.0  00.0  00.0  00.0 00.0
task stealth  5 12  50.0  50.0  50.0  50.0 00.0
task stealth  5 13  00.0  00.0  00.0  00.0 00.0
task stealth  5 14  00.0  00.0  00.0  00.0 00.0
task stealth  5 15  00.0  00.0  00.0  00.0 00.0
task stealth  5 16  00.0  00.0  00.0  00.0 00.0
task stealth  5 17  00.0  00.0  00.0  00.0 00.0
task stealth  5 18  00.0  00.0  00.0  00.0 00.0
task stealth  5 19  00.0  00.0  00.0  00.0 00.0
task stealth  5 20  00.0  00.0  00.0  00.0 00.0
task stealth  5 21  00.0  00.0  00.0  00.0 00.0
task stealth  5 22  00.0  00.0  00.0  00.0 00.0
task stealth  5 23  00.0  00.0  00.0  00.0 00.0
task stealth  5 24  50.0  50.0  50.0   5.0 00.0
task attack  5  1  1 1 1 1 1 1
#
# task 6:  ground contact (Frog launchers)
task general 6  G  FG  .99 0.211  2 frog.icon
task mean  6 10.0  20.0  0.00  0.00  0.00  0.00  0.00  5.00  0.00 1.00
task view  6  2  2  2 2 2 2
task stealth  6  0 100.0 100.0 100.0 100.0 00.0
task stealth  6  1  00.0  00.0  00.0   5.0 00.0
task stealth  6  2  00.0  00.0  00.0   5.0 00.0
task stealth  6  3  00.0  00.0  00.0   8.0 00.0
task stealth  6  4  00.0  00.0  00.0  00.0 00.0
task stealth  6  5  00.0  00.0  00.0  00.0 00.0
task stealth  6  6  00.0  00.0  00.0   6.0 00.0
task stealth  6  7  00.0  00.0  00.0   5.0 00.0
task stealth  6  8  00.0  00.0  00.0  00.0 00.0
task stealth  6  9  00.0  00.0  00.0  00.0 00.0
task stealth  6 10 100.0  00.0  00.0  00.0 00.0
task stealth  6 11  00.0  00.0  00.0  00.0 00.0
task stealth  6 12  00.0  00.0  00.0  50.0 00.0
task stealth  6 13  00.0  00.0  00.0  00.0 00.0
task stealth  6 14  00.0  00.0  00.0  00.0 00.0
task stealth  6 15  00.0  00.0  00.0  00.0 00.0
task stealth  6 16  00.0  00.0  00.0  00.0 00.0
task stealth  6 17  00.0  00.0  00.0  00.0 00.0
task stealth  6 18  00.0  00.0  00.0  00.0 00.0
task stealth  6 19  00.0  00.0  00.0  00.0 00.0
task stealth  6 20  00.0  00.0  00.0  00.0 00.0
task stealth  6 21  00.0  00.0  00.0  00.0 00.0
task stealth  6 22  00.0  00.0  00.0  00.0 00.0
task stealth  6 23  00.0  00.0  00.0  00.0 00.0
task stealth  6 24  50.0  50.0  50.0  00.0 00.0
task attack  6  1  1 1 1 1 1 1
#
```

```
# task 7: ground contact (silkworm anti-ship missile battery)
task general 7  G  SWG  0.99  0.211  1  silkworm.icon
task mean  7  15.0  20.0  0.00  0.00  0.00  0.00  0.00  5.00  0.00 1.00
task view   7 2 2 2  2 2 2
task stealth  7  0 100.0 100.0 100.0  0.0 0.0
task stealth  7  1   0.0   0.0   0.0  5.0 0.0
task stealth  7  2   0.0   0.0   0.0  5.0 0.0
task stealth  7  3   0.0   0.0   0.0  8.0 0.0
task stealth  7  4   0.0   0.0   0.0  0.0 0.0
task stealth  7  5   0.0   0.0   0.0  0.0 0.0
task stealth  7  6   0.0   0.0   0.0  6.0 0.0
task stealth  7  7   0.0   0.0   0.0  5.0 0.0
task stealth  7  8   0.0   0.0   0.0  0.0 0.0
task stealth  7  9   0.0   0.0   0.0  0.0 0.0
task stealth  7 10  50.0  50.0  50.0 00.0 0.0
task stealth  7 11   0.0   0.0   0.0  0.0 0.0
task stealth  7 12   0.0   0.0   0.0  0.0 0.0
task stealth  7 13   0.0   0.0   0.0  0.0 0.0
task stealth  7 14 100.0   0.0   0.0  0.0 0.0
task stealth  7 15 100.0   0.0   0.0  0.0 0.0
task stealth  7 16 100.0   0.0   0.0  0.0 0.0
task stealth  7 17 100.0   0.0   0.0  0.0 0.0
task stealth  7 18 100.0   0.0   0.0  0.0 0.0
task stealth  7 19 100.0   0.0   0.0  0.0 0.0
task stealth  7 20 100.0   0.0   0.0  0.0 0.0
task stealth  7 21 100.0   0.0   0.0  0.0 0.0
task stealth  7 22 100.0   0.0   0.0  0.0 0.0
task stealth  7 23 100.0   0.0   0.0  0.0 0.0
task stealth  7 24 100.0   0.0   0.0  0.0 0.0
task stealth  7 25 100.0   0.0   0.0  0.0 0.0
task attack  7  1  1 1 1 1 1 1
#
# task 8: ground contact (mines)
task general 8  G MN  0.01  0.211  2 mines.icon
task mean  8  5.0  10.0  0.00  0.00  0.00  0.00  2.00  0.00  0.00 1.00
task view  8 2  2  2  2 2 2
task stealth  8  0  100.0 100.0 100.0 100.0 02.5
task stealth  8  1    0.0   0.0   0.0   0.0 0.0
task stealth  8  2    0.0   0.0   0.0   0.0 0.0
task stealth  8  3    0.0   0.0   0.0   0.0 0.0
task stealth  8  4    0.0   0.0   0.0   0.0 0.0
task stealth  8  5    0.0   0.0   0.0   0.0 0.0
task stealth  8  6    0.0   0.0   0.0   0.0 0.0
task stealth  8  7    0.0   0.0   0.0   0.0 0.0
task stealth  8  8    0.0   0.0   0.0   0.0 0.0
task stealth  8  9   00.0  00.0  00.0   8.0 02.0
task stealth  8 10    0.0   0.0   0.0   0.0 0.0
task stealth  8 11    0.0   0.0   0.0   0.0 0.0
task stealth  8 12    0.0   0.0   0.0   0.0 0.0
task stealth  8 13    0.0   0.0   0.0   0.0 0.0
task stealth  8 14    0.0   0.0   0.0   0.0 0.0
```

```
task stealth  8 15    0.0   0.0   0.0   0.0 0.0
task stealth  8 16    0.0   0.0   0.0   0.0 0.0
task stealth  8 17    0.0   0.0   0.0   0.0 0.0
task stealth  8 18    0.0   0.0   0.0   0.0 0.0
task stealth  8 19    0.0   0.0   0.0   0.0 0.0
task stealth  8 20    0.0   0.0   0.0   0.0 0.0
task stealth  8 21    0.0   0.0   0.0   0.0 0.0
task stealth  8 22    0.0   0.0   0.0   0.0 0.0
task stealth  8 23    0.0   0.0   0.0   0.0 0.0
task stealth  8 24    4.0   4.0   4.0   0.0 01.0
task attack 8  1 1 1 1 1 1 1
#
# task 9: sea contact (mines)
task general 9  S MS  0.01 0.211  2 mines.icon
task mean  9  5.0  10.0 0.00  0.00  0.00  0.00  2.00  0.00  0.00 1.00
task view  9 2  2  2  2 2 2
task stealth  9  0 100.0 100.0 100.0 100.0 02.5
task stealth  9  4 6.0   5.0   3.0  10.0 00.0
task stealth  9 14 6.0   5.0   3.0  00.0 2.0
task stealth  9 15 6.0   5.0   3.0  00.0 2.0
task stealth  9 16 6.0   5.0   3.0  00.0 2.0
task stealth  9 17 6.0   5.0   3.0  00.0 2.0
task stealth  9 18 6.0   5.0   3.0  00.0 2.0
task stealth  9 19 6.0   5.0   3.0  00.0 2.0
task stealth  9 20 6.0   5.0   3.0  00.0 2.0
task stealth  9 21 6.0   5.0   3.0  00.0 2.0
task stealth  9 22 6.0   5.0   3.0  00.0 2.0
task stealth  9 23 6.0   5.0   3.0  00.0 2.0
task attack  9 0  1 1 1 1 1 1
#
# task 10: air contacts (sea attackers)
task general 10  A  AS  0.4  0.211  1 air.icon
task mean  10 15.0  20.0  5.00  0.00  0.00  0.00  0.00  0.00  0.00 1.00
task view   10 2 2 2 2 2 2
task attack  10  0  1 0 0 0 1 1
#
#task 11: air contacts (ground attackers)
task general 11  A  AG  0.2  0.127  1  air.icon
task mean 11  4.0  20.0  5.00  0.00  0.00  0.00  0.00  0.00  0.00 1.00
task view   11  1 1 1 1 1 1
task attack 11  1  1 1 1 1 1 1   ·
#
# task 12: air contacts (enemy helo)
task general 12  A  HH .18  0.211  1  helo.icon
task mean  12 15.0  10.0  6.00  0.00  0.00  0.00  0.00  0.00  0.00 1.00
task view  12  2  2  2 2 2 2
task attack  12  0  1 1 1 1 1 1
#
# task 13: air contacts (neutrals)
task general 13  A  NU  0.3  0.211  0  air.icon
task mean  13 10.0  10.0 2.00  0.00  0.00  0.00  0.00  0.00  0.00 0.00
```

```
task view   13 2 2 2 2 2 2
task attack 13  0  1 1 1 1 1 1
#
# task 14: ground contacts (enemy tanks, vehicles)
task general 14  G  TN  0.02  0.211  2  tank.icon
task mean 14 5.0  10.0  0.00  0.00  0.00  0.00  0.00  10.00  0.00 1.00
task view  14 2  2  2  2  2  2
task stealth  14  0  100.0 100.0 100.0 100.0 02.5
task stealth  14  1   0.0   0.0   0.0   0.0 00.0
task stealth  14  2   0.0   0.0   0.0   0.0 00.0
task stealth  14  3  10.0  10.0  10.0   8.0 00.0
task stealth  14  4   0.0   0.0   0.0   0.0 00.0
task stealth  14  5   0.0   0.0   0.0   0.0 00.0
task stealth  14  6  10.0  10.0  10.0   6.0 00.0
task stealth  14  7   0.0   0.0   0.0   0.0 00.0
task stealth  14  8   0.0   0.0   0.0   0.0 00.0
task stealth  14  9   0.0   0.0   0.0   0.0 00.0
task stealth  14 10   5.0   5.0   5.0   0.0 02.5
task stealth  14 11   0.0   0.0   0.0   0.0 00.0
task stealth  14 12   0.0   0.0   0.0  50.0 02.5
task stealth  14 13   0.0   0.0   0.0   0.0 00.0
task stealth  14 14   0.0   0.0   0.0   0.0 00.0
task stealth  14 15   0.0   0.0   0.0   0.0 00.0
task stealth  14 16   0.0   0.0   0.0   0.0 00.0
task stealth  14 17   0.0   0.0   0.0   0.0 00.0
task stealth  14 18   0.0   0.0   0.0   0.0 00.0
task stealth  14 19   0.0   0.0   0.0   0.0 00.0
task stealth  14 20   0.0   0.0   0.0   0.0 00.0
task stealth  14 21   0.0   0.0   0.0   0.0 00.0
task stealth  14 22   0.0   0.0   0.0   0.0 00.0
task stealth  14 23   0.0   0.0   0.0   0.0 00.0
task stealth  14 24   8.0   6.0   6.0   3.0 02.5
task attack 14 1  0 1 1 1 1 1
#
# task 15: ground contacts (neutrals)
task general 15  G  NU  0.05  0.211  0
task mean 15 10.0  10.0  0.00  0.00  2.00  0.00  0.00  0.00  0.00 0.00
task view  15 2 2 2  2 2 2
task attack  15  0  0 0 0 0 0 0
#
# task 16: sea contact (Patrol boats)
task general 16  S  PB  .05 0.211  1  ship.icon
task mean 16 15.0  10.0  0.00  6.00  0.00  0.00  0.00  0.00  0.00 1.00
task view  16  2  2  2 2 2 2
task attack  16  0  1 0 0 0 1 1
#
# task 17: sea contacts (enemy submarines)
task general 17  S  SS  0.05  0.211  1  sub.icon
task mean 17 15.0  10.0  0.00  5.00  0.00  0.00  0.00  0.00  0.00 1.00
task view  17 2  2  2  2  2  2
task stealth  17  0  100.0 100.0 100.0 100.0 02.5
```

```
task stealth  17  1  10.0  10.0  10.0   0.0 00.0
task stealth  17  2  10.0  10.0  10.0   0.0 00.0
task stealth  17  3  10.0  10.0  10.0   0.0 00.0
task stealth  17  4  10.0  10.0  10.0   0.0 00.0
task stealth  17  5  20.0  20.0  20.0  10.0 00.0
task stealth  17  6  10.0  10.0  10.0   0.0 00.0
task stealth  17  7  10.0  10.0  10.0   0.0 00.0
task stealth  17  8  10.0  10.0  10.0   0.0 00.0
task stealth  17  9  10.0  10.0  10.0   0.0 00.0
task stealth  17 10  30.0  10.0  10.0   0.0 00.0
task stealth  17 11  00.0  00.0  00.0   0.0 00.0
task stealth  17 12  00.0  00.0  00.0   0.0 00.0
task stealth  17 13  00.0  00.0  00.0   0.0 00.0
task stealth  17 14  35.0  15.0   7.0   6.0 2.0
task stealth  17 15   8.0   8.0   7.0  00.0 2.0
task stealth  17 16  10.0   9.0   7.0  00.0 2.0
task stealth  17 17   8.0   8.0   7.0  00.0 2.0
task stealth  17 18   8.0   8.0   7.0  00.0 2.0
task stealth  17 19   8.0   8.0   7.0  00.0 2.0
task stealth  17 20   8.0   8.0   7.0  00.0 2.0
task stealth  17 21   8.0   8.0   7.0  00.0 2.0
task attack  17  0  1 0 0 0 1 1
#
# task 18: sea contact (Sea-born Anti-ship cruise missiles)
task general 18  S  ML  0.99  0.211  1
task mean 18  15.0  10.0  0.00  5.00  0.00  0.00  0.00  0.00  0.00 1.00
task view 18  2  2  2 2 2 2
task attack  18  0  1 0 0 0 1 1
#
# task 19: sea contacts (neutrals)
task general 19  S  NU  0.05  0.211  0  ship.icon
task mean 19 10.0  10.0  0.00  2.00  0.00  0.00  0.00  0.00  0.00 0.00
task view  19 2 2 2  2 2 2
task attack  19  0  0 0 0 0 1 1
#
# task 20: Medivac mission
task general 20  G  MV  0.00  0.211  9 cross.icon
task mean  20  5.0  10.0  0.00  0.00  0.00  0.00  0.00  0.00  5.00 0.00
task view  20  2  2  2 2 2 2
task stealth  20  20 100.0 100.0 100.0 100.0  0.0
task stealth  20  24 100.0 100.0 100.0 100.0  0.0
task attack  20  0  1 1 1 1 1 1
#
# task 21: Swamp
task general 21  G  SM .00  0.211  2  swamp.icon.no_label
task mean  21  2.0  10.0  0.00 0.00  20.00  0.00  0.00  0.00  0.00 0.00
task view  21  2  2  2 2 2 2
task stealth  21  0 100.0 100.0 100.0 00.0 0.0
task stealth  21 10 100.0 100.0 100.0 00.0 0.0
task stealth  21 14 100.0 100.0 100.0 00.0 0.0
task stealth  21 15 100.0 100.0 100.0 00.0 0.0
```

145

```
task stealth  21 16 100.0 100.0 100.0 00.0 0.0
task stealth  21 17 100.0 100.0 100.0 00.0 0.0
task stealth  21 18 100.0 100.0 100.0 00.0 0.0
task stealth  21 19 100.0 100.0 100.0 00.0 0.0
task stealth  21 20 100.0 100.0 100.0 00.0 0.0
task stealth  21 21 100.0 100.0 100.0 00.0 0.0
task stealth  21 24 100.0 100.0 100.0 00.0 2.0
task stealth  21 25 100.0 100.0 100.0 00.0 0.0
task attack  21  1  1 1 1 1 1 1
#
# task 22: Air contact (silkworm anti-ship missile battery)
task general 22  A  SWA  0.99  0.211  2  silkworm.icon
task mean  22  15.0  20.0  0.00  0.00  0.00  0.00  0.00  5.00  0.00 1.00
task view   22 2 2 2  2 2 2
task stealth 22  0 100.0 100.0 100.0 100.0  0.0
task stealth 22  1   0.0   0.0   0.0   0.0 0.0
task stealth 22  2   0.0   0.0   0.0   0.0 0.0
task stealth 22  3   0.0   0.0   0.0   0.0 0.0
task stealth 22  4   0.0   0.0   0.0   0.0 0.0
task stealth 22  5   0.0   0.0   0.0   0.0 0.0
task stealth 22  6   0.0   0.0   0.0   0.0 0.0
task stealth 22  7   0.0   0.0   0.0   0.0 0.0
task stealth 22  8   0.0   0.0   0.0   0.0 0.0
task stealth 22  9   0.0   0.0   0.0   0.0 0.0
task stealth 22 10  50.0  50.0  50.0   0.0 0.0
task stealth 22 11   0.0   0.0   0.0   0.0 0.0
task stealth 22 12   0.0   0.0   0.0   0.0 0.0
task stealth 22 13   0.0   0.0   0.0   0.0 0.0
task stealth 22 14 100.0  50.0  50.0   0.0 0.0
task stealth 22 15 100.0  50.0  50.0   0.0 0.0
task stealth 22 16 100.0  50.0  50.0   0.0 0.0
task stealth 22 17 100.0  50.0  50.0   0.0 0.0
task stealth 22 18 100.0  50.0  50.0   0.0 0.0
task stealth 22 19 100.0  50.0  50.0   0.0 0.0
task stealth 22 20 100.0  50.0  50.0   0.0 0.0
task stealth 22 21 100.0  50.0  50.0   0.0 0.0
task stealth 22 22 100.0  50.0  50.0   0.0 0.0
task stealth 22 23 100.0  50.0  50.0   0.0 0.0
task stealth 22 24 100.0  50.0  50.0   0.0 0.0
task stealth 22 25 100.0  50.0  50.0   0.0 0.0
task attack 22  1  1 1 1 1 1 1
#
# task 23: Dummy task with no value
task general 23  A  DUM  0.99  0.211  2
task mean  23  0.0  0.0 0.00  0.00  0.00  0.00  0.00  0.00  0.00 0.00
task view   23 2 2 2  2 2 2
task attack 23  0  0 0 0 0 0 0
#---------
task penetration 0  0 0 0 0 0 0 0
task penetration 1  0 0 0 0 0 0 0
task penetration 2  0 0 0 0 0 0 0
```

146

task penetration 3 0 0 0 0 0 0 0
task penetration 4 0 0 0 0 0 0 0
task penetration 5 0 0 1 1 0 0 0
task penetration 6 0 0 1 1 0 0 0
task penetration 7 0 0 0 0 0 1 1
task penetration 8 0 0 0 0 0 0 0
task penetration 9 0 0 0 0 0 0 0
task penetration 10 0 0 0 0 0 1 1
task penetration 11 0 0 1 1 0 0 0
task penetration 12 0 0 0 0 0 1 1
task penetration 13 0 0 0 0 0 0 0
task penetration 14 0 0 0 0 0 0 0
task penetration 15 0 0 0 0 0 0 0
task penetration 16 0 0 0 0 0 1 1
task penetration 17 0 0 0 0 0 1 1
task penetration 18 0 0 0 0 0 1 1
task penetration 19 0 0 0 0 0 0 0
task penetration 20 0 0 0 0 0 0 0
task penetration 21 0 0 0 0 0 0 0
task penetration 22 0 0 0 0 0 1 1
#
end_of_class_info
#
#<<<<<<<<<<<<<<<< State Information >>>>>>>>>>>>>>>
#
platform number 11
----
platform definition 0 CVN   4 70.00 15.00
platform definition 1 CG   4 88.00 54.00
platform definition 2 FFG   4 90.00 50.00
platform definition 3 DD1   5 40.00 73.00
platform definition 4 DD2   5 40.00 78.00
platform definition 5 LA   5 64.00 73.00
platform definition 6 LP   5 64.00 77.00
platform definition 7 LH   5 64.00 75.00
platform definition 8 LC1   2 60.00 73.00
platform definition 9 LC2   3 60.00 75.00
platform definition 10 BAS 0 96.00 04.00
#platform definition 11 GOD 0 50.00 50.00
#
#---------
#TASK DEFINITIONS
#-------------------------------------
#
Tasks 200-225 (swamp areas)
#
task definition 200 21 1 0
#
task definition 201 21 1 0
#
task definition 202 21 1 0

147

```
#
task definition  203  21 1 0
#
task definition  204  21 1 0
#
task definition  205  21 1 0
#
task definition  206  21 1 0
#
 task definition  207  21 1 0
#
 task definition  208  21 1 0
#
task definition  209  21 1 0
#
 task definition  210  21 1 0
#
 task definition  211  21  1 0
#
task definition  212 21 1 0
#
 task definition  213  21 1 0
#
 task definition  214  21 1 0
#
 task definition  215  21 1 0
#
 task definition  216  21 1 0
#
 task definition  217  21 1 0
#
 task definition  218  21 1 0
#
 task definition  219  21 1 0
#
task definition  220  21 1 0
#
 task definition  221  21 1 0
#
 task definition  222  21 1 0
#
 task definition  223  21 1 0
#
 task definition  224  21 1 0
#
 task definition  225  21 1 0
#
#----------------------------
# Tasks 226-230 (ground mines)
#
 task definition  226  23  1 0
```

```
#
 task definition  227  23  1 0
#
 task definition  228  23 1 0
#
 task definition  229  23 1 0
#
 task definition  230  23 1 0
#
#----------------------------------------
# Tasks 231-235 (ground contacts - enemy)
 task definition  231  23  1 0
#
 task definition  232  23 1 0
#
 task definition  233  23 1 0
#
 task definition  234  23 1 0
#
 task definition  235  23 1 0
#
#-----------------------------
# Tasks 236-240 (sea mines)
#
 task definition  236  9  0 0
#
 task definition  237  9  0 0
#
 task definition  238  9  0 0
#
 task definition  239  23  0 0
#
 task definition  240  23  0 0
#
#-----------------------------
# Tasks 241-250 (artillery)
#
task definition 241 5 1 0
#
task definition 242 5 1 0
#
task definition 243 5 1 0
#
task definition 244 5 1 0
#
task definition 245 5 1 0
#
task definition 246 5 1 0
#
task definition 247 5 1 0
#
```

149

```
task definition 248 5 1 0
#
task definition 249 5 1 0
#
task definition 250 5 1 0
#
#---------------------------
# Tasks 251-255 (ground contacts - enemy)
#
task definition 251 23 1  0
#
task definition 252 23 1  0
#
task definition 253 23 1  0
#
task definition 254 23 1  0
#
task definition 255 23 1  0
#
#---------------------------------
# Tasks 256 - 260 (ground mines)
#
task definition 256 8 1  0
#
task definition 257 8 1  0
#
task definition 258 23 1  0
#
task definition 259 23 1  0
#
task definition 260 23 1  0
#
#----------------------------
# Tasks 261-265 (aircontacts, anti-shipping)
#
 task definition  261  10 0 1
#
 task definition  262  10 0 1
#
 task definition  263  23 0 1
#
 task definition  264  23 0 1
#
 task definition  265  23 0 1
#
#----------------------------
# Tasks 266-270 (aircontacts, anti-ground)
#
 task definition  266  23 1 1
#
 task definition  267  23 1 1
```

```
#
 task definition  268  23  1  1
#
 task definition  269  23  1  1
#
 task definition  270  23  1  1
#
#----------------------------------
# Task 271-275  (air contacts - helicopter)
#
 task definition  271  12  0  3
#
 task definition  272  12  0  3
#
 task definition  273  23  0  3
#
 task definition  274  23  0  3
#
 task definition  275  23  0  3
#
#----------------------------------
# Task 276-280  (air contacts - neutral)
#
 task definition  276  13  0  3
#
 task definition  277  23  0  3
#
 task definition  278  23  0  3
#
 task definition  279  23  0  3
#
 task definition  280  23  0  3
#
#----------------------------------
# Task 281-285 (ground contacts - frog launchers)
#
 task definition  281  6  1  3
#
 task definition  282  23  1  3
#
 task definition  283  23  1  3
#
 task definition  284  23  1  3
#
 task definition  285  23  1  3
#
#----------------------------------
# Task 286-290 (ground contacts - silkworm launchers)
#
 task definition  286  7  1  3  #(ground)
#
```

```
 task definition  287 23 1 3
 #
 task definition  288 23 1 3
 #
 task definition  289 22 1 3 #(air)
 #
 task definition  290 23 1 3
 #
#----------------------------------
# Task 291-295 (medivac  mission)
 #
 task definition  291  23 0 3
 #
 task definition  292  23 0 3
 #
 task definition  293  23 0 3
 #
 task definition  294  23 0 3
 #
 task definition  295  23 0 3
 #
#----------------------------------
# Task 296-300 (sea contacts - submarine)
 #
 task definition  296 17 0 3
 #
 task definition  297 17 0 3
 #
 task definition  298 23 0 3
 #
 task definition  299 23 0 3
 #
 task definition  300 23 0 3
 #
#----------------------------------
# Task 301-305 (sea contacts - patrol boats)
 #
 task definition  301 16 0 3
 #
 task definition  302 16 0 3
 #
 task definition  303 23 0 3
 #
 task definition  304 23 0 3
 #
 task definition  305 23 0 3
 #
#----------------------------------
# Tasks 306-314 (ground missions)
 #
task definition  306  0 1 0  #(hill)
```

```
#
task definition  307  2 1 0 #(port)
#
task definition  308  1 1 0 #(airport)
#
task definition  309   4 1  0 #(taking beach one)
#
task definition  310  4 1 0 #(taking beach two)
#
task definition  311  3 1 0
#
task definition  312 23 1 0
#
task definition  313 23 1 0
#
task definition  314 23 1 0
#
#----------------------------
# Tasks 315-319 (ground mines)
#
 task definition  315  23  1 0
#
 task definition  316  23  1 0
#
 task definition  317  23 1 0
#
 task definition  318  23 1 0
#
 task definition  319  23 1 0
#
#------------------------------------
# Tasks 320- 324 (ground contacts - enemy)
 task definition  320  14  1 0
#
 task definition  321  14 1 0
#
 task definition  322  23 1 0
#
 task definition  323  23 1 0
#
 task definition  324  23 1 0
#
#--------------------------------
# Task 325-329 (sea contacts - neutrals)
#
 task definition  325  19 0 3
#
 task definition  326  23 0 3
#
 task definition  327  23 0 3
#
```

```
 task definition  328  23 0 3
 #
 task definition  329  23 0 3
 #
#--------------------------------
# Task 330-334  (ground contacts - silkworm launchers)
 #
 task definition  330  23 1 3
 #
 task definition  331  23 1 3
 #
 task definition  332  23 1 3
 #
 task definition  333  23 1 3
 #
 task definition  334  23 1 3
 #
#------------------------------
# Task 335-339 (medivac mission)
 #
 task definition  335  20 0 3
 task attribute  335
  10.0 20.0 0.00 0.00 0.00 0.00  0.00  0.00 5.00 0.00
 #
 task definition  336  20 0 3
 #
 task definition  337  23 0 3
 #
 task definition  338  23 0 3
 #
 task definition  339  23 0 3
 #
#---------
 #
task prerequisites  310 3  236  237  238
 #
#---------
 #
task spawn  306  1  A  311
task spawn  310  4  A  320   321  335  336
task spawn  286  1  D  289
 #
#<<<<<<<<<<<<<<<< Maneuver Information >>>>>>>>>>>>>>
NOTE - SPECIFIC EXAMPLES ONLY OF MANEUVERS PROVIDED
# Tasks 200-220 (swamp areas)
 #
maneuver definition   200  1.000
s 18.0  89.0  3600.0
e 18.0  89.0  0.0
 #
#----------------------------
```

154

```
# Tasks 226-230 (ground mines)
#
maneuver definition   226  6000.000
s  15.0   60.0 3600.0
e  15.0   60.0  0.0
#
#------------------------------
# Tasks 236-240 (sea mines)
#
maneuver definition   236  10.00
s  33.0   80.5  3600.0
e  33.0   80.5  0.0
#
#------------------------------
# Tasks 241-250 (artilery)
#  Artillery positions:  site one: (12.0  67.0 - north)
#                 site two: (15.0  88.0 - south)
#
maneuver definition   241  500.0
s  12.0  67.0  300.0
m  12.0  67.0  0.99
e  27.5  72.5  0.0
#
#----------------------------
# Tasks 251-255 (ground contacts - enemy)
#
maneuver definition   251  6000.00
s  22.0   70.0 3600.0
e  22.0   70.0  0.0
#
#----------------------------
# Tasks 261-265 (aircontacts, anti-shipping)
#
maneuver definition   261  900.00
m  1.00   20.0  0.9
e  75.0   20.0  0.0
#
#----------------------------
# Tasks 266-270 (aircontacts, anti-ground)
#
maneuver definition   266  6000.000
s  10.0   70.0 3600.0
e  10.0   70.0  0.0
#
#----------------------------------
# Task 271-275  (air contacts - helicopter)
#
maneuver definition   271  600.000
m  5.0   95.0  0.9
e  64.0   75.0  0.0
#
```

```
#----------------------------------
# Task 276-280  (air contacts - neutral)
#
maneuver definition  276 6.000
m  10.0   70.0 0.50
e 99.0   70.0 0.0
#
#----------------------------------
# Task 281-285 (ground contacts - frog launchers)
#
maneuver definition  281 720.000
s 3.00   85.0  1200.0
m 3.00   85.0  0.88
e 28.0   83.0  0.00
#
#----------------------------------
# Task 286-290 (ground contacts - silkworm launchers)
#
maneuver definition  286  1200.000
s 28.5  51.5  720.0
e 28.5  51.5   0.0
#
#----------------------------------
# Task 291-295 (medivac  mission)
#
maneuver definition  291  6000.000
s 28.0  73.0 3600.0
e 28.0  73.0  0.0
#
#----------------------------------
# tasks 296-300  (sea contacts - submarines)
#
maneuver definition  296  120.000
m  99.0   40.0  0.9
e 75.0   20.0  0.0
#
#----------------------------------
# Task 301-305 (sea contacts - patrol boats)
#
maneuver definition  301  600.000
m  32.0   38.0  0.99
e 70.0   20.0  0.0
#
#----------------------------------
# tasks 306-314  (ground missions)
#
maneuver definition  306  1.000  #(hill)
s 25.00  60.00  3600.0
e 25.00  60.00  0.00
#
```

APPENDIX B [DDD Tutorial]

1. The DDD Graphical User interface

The DDD graphical user interface displays a map on the left side of the screen which is a graphical representation of friendly and enemy objects. Within the map, land is represented by squares which have a brown tint, and sea by squares which are white. The mouse commands listed in the next section will describe how friendly objects on the map may be manipulated and how information on enemy objects is obtained. The right half of the screen contains four buttons:

Start/Refresh: The Start button is used only at the beginning of a scenario to start all of the stations playing. Once the scenario has begun, the button changes to Refresh. Left clicking on the Refresh button redraws the map eliminating any undesired traces which may appear.

Zoom In: Allows the user to zoom in for a more detailed look at a particular section of the map. To zoom in, left click on the "Zoom In" button. Move the cursor over to map and left click at a point to the left or right of where the area of interest lies. While continuing to hold the left mouse button depressed, drag the cursor and a box will begin to appear showing the area which will be zoomed in on.

Zoom Out: Left clicking on this button returns the map to the previous map size.

Cancel: Left Clicking on the Cancel button allows the user to suspend an operation on an asset such as a move or an attack prior to completing the task.

The right half of the screen also contains a time bar. When a friendly object (platform or sub-platform) is selected to perform an action (i.e. launch aircraft, attack a task), a white arrow will appear next to this bar showing the amount of time to complete this task. The object cannot perform any other action until this action is completed. In addition, above the time bar are several other pieces of information. The color of the stick man figure in a box shows the color of the objects on the map which your station controls. Next to the this box is the name of the station you are playing (i.e. CJTF, MEU 2, etc.) Below the box are two counters which display feedback on how well the entire team is doing on the scenario. The counter labeled mission starts at zero and increments as tasks are accomplished. The counter labeled strength starts at 100 and decrements as your force strength is decreased.

The lower portion of the screen contains two window dialog boxes. Close attention must be given to the window on the left as this box displays messages between the various players which may require some action to be taken by your station. The right window can

157

best be described as a confirmation window. Summaries of messages or actions performed by your station will appear in this window along with some messages about the status of other friendly objects. Also, the very bottom of the screen below these two windows displays warning and error prompts. A beep will occur along with a warning or error message following any action performed by your station which is not allowed (i.e. Attempting to attack the enemy when your unit is out of range).

2. Using the Mouse in DDD

A standard three button mouse is used when running a scenario at each workstation. When clicking on an object in DDD each mouse button serves a different function depending on whether the object is friend or foe, and if friend whether or not the object is owned or not owned by you.

Left Mouse Button: The left mouse button clicked on an object will just select it. The left mouse button is also used to carry out options selected from the menu presented when an object is right clicked on.

Middle Mouse Button: When the middle button is clicked on an object, the window presented depends on whether the object is a (1) friendly platform or sub-platform or (2) enemy platform or task. If the object is an enemy platform or task , a window appears which provides known information about the attributes of the object. If a friendly object is selected, a screen appears which displays the attributes of that platform or sub-platform. If a platform is selected the attributes, ownership, and number of all sub-platform located on the platform are also shown. Platforms are the major friendly objects in the scenarios. Sub-platforms are objects such as aircraft, Naval Surface Fire Support missions, standard missile missions, helicopters, etc. which are carried by a platform. The ownership of any sub-platform may or may not be the same as the owner of the platform it is being carried on.

When a friendly platform is selected with the middle mouse button, the portion of the window where the sub-platforms are listed is used to launch or request launch of a sub-platforms depending on where the sub-platform is located. There are three possible situations which can occur here:

1. Sub-platform needed to be launched is on a platform owned by you: In this case you can launch any sub-platform on your platform whether you own it or not. This is done by left clicking on the right arrow key in the line for the number of sub-platform(s) needed and then clicking OK..

2. Sub-platform needed to be launched and which is owned by you is located on a platform which you do not own: In this case middle click on the platform where

your sub-platform is located. Left click on the arrow located in the line of the sub-platform needed until the desired number of sub-platforms to be launched is set, and then left click on OK. A message will then be sent to the owner of the platform where your sub-platform is located requesting that it be launched. It is the responsibility of the person where your sub-platform resides to launch it.

3. Sub-platform needed is not owned by you and is located on a platform not owned by you: In this case middle click on the platform where the sub-platform needed resides. Left click on the arrow on the line containing the sub-platform desired until the required number is set, and then left click on OK. A message will then be forwarded up your chain of command which must be acted upon by your immediate superior to obtain this sub-platform.

The lower portion of this screen also offers options for displaying information on range rings for both sensors and weapons of a friendly object against either enemy ground, air, or sea assets. The sensor option will display four range rings around the object. The outermost black ring represents the detection range; the next inner black ring represents the range at which measurements on the enemy can be made; the furthest inner black ring represents the visual detection range; and the inner yellow ring represents the range at which the object is vulnerable. The weapons option displays a single red ring around the object which shows the effective range of its weapon. To display these range rings left click on either sensors, weapon, or both, and then left click what type of medium to display these for (air, ground, or sea). To turn the range rings off, middle click on the object and left click on none.

Right Mouse Button: The right mouse button will cause a menu to pop up with the menu options depending on ownership of the object and whether it is a friendly or enemy object or task. The following sections describe the options presented depending on the object selected with the right mouse button.

1. Friendly object which you do not own: The menu that pops up will present the option of requesting the asset, forcing the transfer of the asset, or information on the asset. Explanations of these options follow:

Request (REQ): A menu will pop which allows selecting who the request is to, and the urgency of the request. All items must be selected. When the choices are completed, a message is sent to the person selected if they are directly in your chain of command or up your chain of command where your superior must take action on the request.

Force Transfer (FORCE XFR): This option may only be used if the station you

are playing is superior to someone else in the scenario (i.e. CJTF, MCC, or GCC). Units may only be forcibly transferred which are under control of a person subordinate to you. When this option is selected, a menu will pop up which lists the other players in the scenario. Left click on the person the asset is desired to be forcibly transferred to and click OK. The object will be transferred to the person selected provided the object is transferable object.

Information (INFO): Same as middle clicking on the object.

<u>Friendly object which you own:</u> This menu will allow the choices of Move, Pursue, Attack, Return, Transfer, or Information. An explanation of these options follow:

Move: Selecting move will cause a cross-hair type symbol to appear. Position this cross hair to the place the object is desired to be moved and single click with the left mouse button. The object will then move to this position. When it arrives there, it will stop until another command to move is given.

Pursue: Selecting pursue will cause the cursor to change to a finger. Place the finger on the enemy object desired to be pursued, left click, and your object will then move to pursue it.

Attack: Attack in DDD does not only have the connotation of physically performing destruction on an object, but also implies that the object selected will carry out a task of some sort. When this option is selected a question mark will appear. Place the question mark on the object or task an action is to be performed on. If in range to perform the this action, a menu will then appear which shows the attributes of the object selected to perform the task and the attributes of the object that the task to be performed on. The option is then given to carry out the task or to cancel the assignment. If the object selected to attack the enemy does not have enough combat power to accomplish the task, a <u>coordinated attack</u> may be performed. It should be noted that the following explanations of how to do a <u>coordinated attack will work only if all of the objects are within attack range</u>. A <u>coordinated attack using two objects</u> is accomplished by first selecting one of the two objects to perform the coordinated attack with the left mouse button, and then right clicking on the second object performing the attack. The menu will then pop up and select the attack option. The cursor will then change to the question mark. Place it on the object which is to be attacked and left click with the mouse. To perform a <u>coordinated attack with three or more objects,</u> left click on the first object performing the attack. Then, while holding the shift key down on the keyboard, left click on all but one of the remaining objects performing the attack. Release the shift key and right click on the final object. The menu will pop up and select attack.

The cursor will change to a question mark. Place it on the object to be attacked and left click.

Return: This option may only be used for sub-platforms. Selecting this option will cause the sub-platform to return to the platform it originated from. The sub-platform will not move towards its originating platform, but instead will change to a box with a "x" in it to simulate returning to its originating platform. The return option has been disabled on some sub-platforms in the scenario. If one of these sub-platforms is directed to return, an error message will appear.

Transfer (XFR): Used to transfer control of an object from one person in the chain of command to another, and to inform another player that an object requested for transfer will not be transferred . When selected, a menu will pop up showing all members of the chain of command. Select the person to whom control is to be passed or to whom a request for transfer is to be denied by left clicking on that person. Then, to transfer control ensure the "transfer right now" button is depressed and click OK; or to send a message denying the request left click on "I can't transfer right now" and click OK. The object will then transfer control to the person selected or a message will be sent informing them that the request for transfer has been denied.

Information (INFO): Same as middle clicking on the object.

Enemy Objects or Tasks: The menu presented in this instance presents the options of Identify, Requesting Information, Transferring Information, Coordinating Action, Assigning, and Information. Explanations of these options follow:

Identify: This option is normally used to identify enemy objects or tasks for which the identity is unknown. This will be readily apparent in a scenario as the first letter shown with the icon will be followed by a question mark. The first letter designates which medium the unknown contact operates in. "A?" implies an unknown air contact; "G?" implies an unknown ground contact; "S?" implies an unknown sea contact. Selecting the identify option will cause a menu to pop up which shows the known attributes of the object or task as seen by each player in the scenario. If a friendly object having sensors capable of identifying the enemy object or task is within sensor range the object will be identified correctly. If not, the question mark will remain. This will be apparent by looking at the lower left hand column where the identity will be shaded from a list of possible identities. Click the fused button near the top left hand corner and then OK. The identity of the object will then appear correctly on the map and its icon will change to its correct identity.

The following tables give descriptions of the two letter symbols which will be the options shown when identifying an object:

Unknown Ground	Description
?	Unknown
HL	Ground mission of taking a hill
AP	Airport ground mission
SP	Seaport ground mission
HD	Holding or occupying ground
TK	Taking a ground mission
AT	Enemy artillery
FG	Enemy Frog launcher
SWG	Enemy Silkworm missile launcher
TN	Enemy tanks, troops, or vehicles
NU	Neutral
MN	Land Mines

Unknown Air	Description
?	Unknown
AS	Enemy attack against ships
AG	Enemy attack against ground force
HH	Enemy helo attack against ships
NU	Neutral
SWA	Silkworm missile in flight

Unknown Sea	Description
?	Unknown
MS	Sea mines
PB	Enemy patrol boats
SS	Enemy submarines
ML	Enemy anti-ship cruise missiles
NU	Neutral

Request Information (REQ INFO): Selecting this option will cause a menu to pop up which allows selecting anther player or all other players you wish information on the enemy object or task from . Select the person(s) and click OK. A message will then be sent to the person(s) notifying them that this information is requested.

Transfer Information (XFR INFO): Selecting this option will cause a menu to pop up which allows selecting a particular individual or all other players you wish information on the enemy object or task to be sent to . Select the person(s) and click OK. A message will then be sent to the person(s) selected.

Coordinate Action (CRD ACTION): The use of this option allows messages to be sent between players concerning action requests, support, or intent against an enemy object or task. When selected, a menu pops up displaying options for choosing who the message is to sent to and a list of messages which may be sent. The following messages may be sent:

1. I plan to handle.
2. I plan to support.
3. I cannot handle.
4. I cannot support.
5. Can you handle ?
6. Can you support ?

Select the person the message is to be sent to, a message is to be sent, and click OK. The message will then be sent to the person selected.

Assign: This option may only be used if you are playing a position where you are superior to someone in the chain of command and may only be directed at those people who are subordinate to you. This option will cause a question mark to appear. Place it on the enemy object or task desired to be assigned and left click. A menu will then appear which allows selecting whom in the chain of command it is to be assigned to. Left click on the person desired to assign the task to and click OK. A message will then be sent to that person notifying them they are responsible for taking care of the task.

Information (INFO): Same as middle clicking on the object.

3.0 List of Objects in the Scenario

Terrain and task objects
    The following shows representations of the icons which represents terrain or task objects in the scenarios.

Swamp: The swamp icon indicates areas which mechanized or infantry units <u>should not traverse</u>. Friendly units will not be destroyed by going into these areas, but total strength will be diminished.

Airfield: The airfield icon is used twice in the scenarios. The enemy has one airfield which is the objective or task to completed by MEU 2. This airfield has attributes associated with it which must be compared to the attacking force attributes to determine if the necessary force is available. The second airfield is friendly owned and controlled by the CJTF. It contains additional assets such as an SR-71 and F-15's which are also under control of the CJTF.

Port: The port is the objective or task of MEU 1. It, like the airfields, also has attributes which must first be determined and compared to attacking forces attributes to determine if enough combat power can be brought to bare to achieve this objective.

Hill: The hill is commanding terrain between the port and airfield which must be captured by MEU 1. It is surrounded by swamps on both sides which means the only way of accomplishing this task is by using heli-borne assault troops (HTP sub-platform).

Task: The task icon has attributes which must first be identified and then a determination made as to the best asset available to complete this task. Tasks are normally

163

used to represent enemy ground forces in a given location which must be eliminated.

✚ Medivac: The medivac icon is a task which may appear after friendly ground platforms or sub-platforms engage enemy objects or tasks. The task has attributes which must be determined. The task is completed by attacking it with the medivac helicopter (HMV sub-platform).

▣▣▣▣▣Hold: The hold icon may appear after completion of a task (i.e. attacking the hill). If this occurs the asset used to perform the task must remain in its current position and may not be used to perform any other task.

Enemy Assets

The following section shows the icons which represent enemy forces that may or may not appear in a scenario. The text which follows each icon describes the enemy objects capabilities and the friendly weapon of choice to use against it.

◣◤ Artillery: Enemy artillery pieces may pop up at various times. When they appear, they take approximately 15 minutes to set up before they are able to fire. They pieces are stored in reinforced concrete bunkers with the ammunition stored in deep underground bunkers. The only method in which enemy artillery may be suppressed is through the use of Naval Surface Fire support (NSFS). Once the artillery pieces begin to move toward you, which simulates firing, you will be unable to attack them.

◈ Mines: The enemy possesses the possibility of deploying both land and sea based mines. If encountered and moved through by friendly forces the total effectiveness of these forces will be diminished. Sea based mines may only be cleared by the use of a mine clearing helicopter (MCM sub-platform) located on the ARG ships. Land mines may only be cleared through the use of the engineering platoon owned by MEU 1 (HE sub-platform).

⊻ Frog Missile sites: These sites are capable of launching short range missiles containing chemical munitions. The launchers take approximately 30 minutes to set up. Naval Surface Fire Support is ineffective at suppressing these launchers. Suppression must be done through the use of Close Air Support (CAS) aircraft carrying precision guided munitions located on the aircraft carrier (VA sub-platform).

◤ Silkworm Missile Site: The enemy has placed silkworm missile sites in residential areas near the port. The appearance of a silkworm site requires visual confirmation through

use of the SR-71 (SR7 sub-platform) prior to attacking the site. The site may only be destroyed by using CAS carrying precision guided munitions (VA sub-platform).

Submarines: The enemy submarines are Kilo class diesel boats. They can only be destroyed using the FFG platform.

Ship: The only ships the enemy possesses are fast patrol boats. These can only be destroyed by using the SH-60 helicopters (H60 sub-platform) which carry penguin missiles.

Helicopter: The enemy possesses Hind helicopters capable of carrying Exocet anti-ship missiles. These helicopters are very low flyers which precludes the use of standard missiles to defend against them. The only friendly asset capable of destroying them are the stinger platoon (SD sub-platform) located on one of the ARG ships.

Aircraft: Enemy aircraft may launch attacks against the ARG or the CVBG. Aircraft may be destroyed by using either standard missiles (SAM sub-platform) located onboard the Aegis cruiser or by using fighter aircraft (VF or F15 sub-platforms) located on the carrier or at the Sigonella airbase.

Tanks: Enemy tanks may be encountered along the road for the assaults on both the airfield and the port. The tanks can only be seen when within the detection range of friendly ground forces. If friendly forces move out of range the tank icon will disappear. Tanks can only be destroyed by using the Cobra helicopters (HCB sub-platform) armed with TOW missiles which are located on the ARG ships.

Unknown Enemy Object: When this icon appears it must first be identified to determine what it is. The icon will have a letter designation followed by a "?". "A" implies unknown air; "G" implies unknown ground; and "S" implies unknown sea. The object must be identified with a suitable friendly platform or sub-platform. Identification of unknown ground objects may only be accomplished using the SR-71 (SR7 sub-platform) located at the Sigonella airbase.

Friendly Assets

Friendly Platform Icon: This icon is used to represent friendly platforms in a scenario. The middle of the box will contain a letter to show the type of medium in which

165

the platform operates. The letter "G" implies a ground asset; the letter "S' implies a sea asset; and the letter "A" implies an air asset. An additional letter and number designator will be shown on the map above the icon for further identification (i.e. CVN-000). Platform icons are color coded to show ownership.
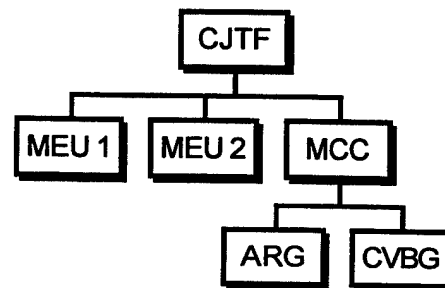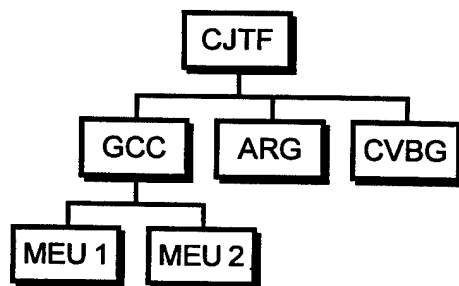
⊗ Friendly Sub-platform Icon: When launched from its parent platform a sub-platform will appear as circle with a letter and number combination above it for further identification (i.e. MCM-101). The middle of the circle will contain a letter to show the type of medium in which the platform operates. The letter "G" implies a ground asset; the letter "S' implies a sea asset; and the letter "A" implies an air asset. Sub-platform icons are also color coded to show ownership.

⊠ Friendly Platform/Sub-platform Busy Icon: When a platform or sub-platform is directed to perform some task such as attacking; transfer ownership between players; launch a sub-platform; or when a sub-platform is directed to return, the icon will change to a box with a "x" in it. The platform or sub-platform cannot be directed to perform any other function until this task is completed. At the end of the task it will change back to its previous form.

Chain of Command

One of the following organizational structures will be used in the running of the scenario depending on which scenario is being run.

```
        CJTF                                    CJTF
         |                                       |
   ┌─────┼─────┐                    ┌─────────┬──┴──────┐
  GCC   ARG  CVBG                 MEU 1     MEU 2     MCC
   |                                                    |
┌──┴──┐                                           ┌─────┴─────┐
MEU 1 MEU 2                                       ARG       CVBG
```

Commander Joint Task Force (CJTF): Overall commander of the operation as delineated in the oporder. The CJTF owns the Sigonella airbase and various sub-platforms in the scenarios. Units controlled by the CJTF in a scenario will be black in color.

Ground Component Commander (GCC): Reports directly to the CJTF when used in the chain of command. Responsible for coordination of the ground assets of MEU 1 and MEU 2 as delineated in the oporder. Units controlled by the GCC will be green in color.

166

Maritime Component Commander (MCC): Reports directly to the CJTF when in the chain of command. Responsible for coordination of maritime assets of the ARG and CVBG as delineated in the oporder. Units controlled by the MCC will be green in color.

Amphibious Ready Group (ARG): The ARG controls a group of three amphibious ships consisting of a LHA, LPD, and a LHD, and two Spruance class destroyers. Units under the ARG's control are orange in color.

Carrier Battle Group (CVBG): The CVBG controls a battlegroup consisting of a Nimitz class carrier , a Ticonderoga class cruiser, and an Perry class frigate. Units under the CVBG's control will be red in color.

Marine Expeditionary Unit (MEU 1 & MEU 2): The two MEU's each own there own LCAC for transporting troops ashore for an amphibious assault. Each LCAC carries the infantry portion of the MEU mounted in AAV's for mobility once they reach the shore. Each MEU also owns helicopter assets which reside on the ARG. Units under MEU 1 are blue in color while those of MEU 2 are pink in color.

LIST OF REFERENCES

1. Kleinman, D. L. and A. Song, "A Research Paradigm for Studying Team Decisionmaking and Coordination," Proc. 1990 JDL Symposium on Command and Control Research, Monterey, CA, June 1990, pp. 129-135.

2. Kleinman, D. L., Serfaty, D., & Luh, P. B. (1984). "A Research Paradigm for Multi-Human Decision Making", Proceedings of the 1984 American Control Conference (pp. 6-11), San Diego, CA.

3. Kleinman, D. L., Luh, P. B., Pattipati, K. R. (1990). "Mathematical Models of Team Distributed Decisionmaking", to appear in: Teams: Their Training and Performance, R. W. Swezey & E. Salas (Eds.), New York: ABLEX, 1990.

4. Michael C. Berigan. *Task Structure and Scenario Design.* Master's Thesis, Naval Postgraduate School, Monterey, California, June 1996.

5. Kemple, Kleinman, Berigan . (1996). "A2C2 Initial Experiment: adaptation of the Joint Scenario and Formalization".

6. Kemple, Hutchins, Kleinman, Sengupta, Berigan & Smith. (1996). "Early Experiences with Experimentation on Dynamic Organizational Structures".

7. Sullivan, (1996). DDD Tutorial for A2C2 phase one experiments conducted March 1996

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center   2
   8725 John J. Kingman Rd., STE 0944
   Ft. Belvior, VA 22060-6218

2. Dudley Knox Library   2
   Naval Postgraduate School
   411 Dyer Rd.
   Monterey, California 93943-5101

3. Professor David Kleinman, Code C3   3
   Naval Postgraduate School
   Monterey California 93943

4. Professor William G. Kemple Code CC   1
   Naval Postgraduate School
   Monterey California 93943

5. Professor Gary Porter Code CC   1
   Naval Postgraduate School
   Monterey California 93943

6. Professor Mike Sovereign Code CC/FM   1
   Naval Postgraduate School
   Monterey California 93940

7. Dr. Elliot E. Entin   2
   Alphatech, INC.
   Executive Place III
   50 Mall Road
   Burlington MA 01803

8. Professor Kathleen Carley   1
   Dept. of Social and Decision Sciences
   Carnegie Mellon University
   Pittsburgh PA 15123

9. Professor John Hollenbeck   1
   Management Department
   Michigan State University
   East Lansing MI 48824

10. Professor Alexander Levis   1
George Mason University
4400 University Drive
Fairfax VA 22030

11. Dr. Willard Vaughan   1
Office of Naval Research
Code 342
800 N. Quincy Street
Arlington VA 22217

12. Dr. Gerald Malecki   1
Office of Naval Research
Code 342
800 N. Quincy Street
Arlington VA 22217

13. Daniel Serfaty   1
APTIMA, Inc.
25 Mall Road
Suite 300
Burlington MA 01803

14 Professor Krishna Pattipati   1
Department of ESE - U157
260 Glenbrook Rd.
Storrs, CT 06269-3157

15. Communications Officer   1
Cruiser Destroyer Group One
PSC 1 Box 2031
APO AA 34001-2031

16. Capt Donald Loren   1
Commander, Destroyer Squadron twenty-eight
Unit 60549
FPO AE 09506-4735